

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2004-258905

(43)Date of publication of application : 16.09.2004

(51)Int.Cl.

G06F 9/32

(21)Application number : 2003-047909

(71)Applicant : JAPAN SCIENCE & TECHNOLOGY AGENCY

(22)Date of filing : 25.02.2003

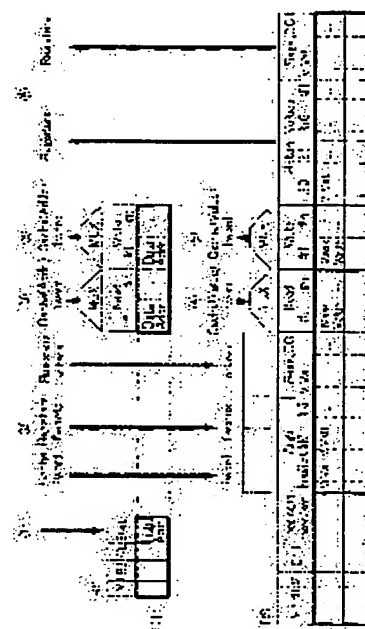
(72)Inventor : NAKAJIMA YASUHIKO

(54) DATA PROCESSING DEVICE AND PROGRAM AND RECORDING MEDIUM FOR RECORDING THIS PROGRAM

(57)Abstract:

PROBLEM TO BE SOLVED: To provide a data processing device for reading a command sequence and/or values from a main storage means to carry out the process of writing the results of operations into the main storage means and capable of increasing the speed of the operations through the reuse of the values even if the command sequence is a loop.

SOLUTION: The start address and end address of a loop to be reused and a main storage address about inputs/outputs are recorded on an RF and information about all registers about the inputs/outputs of the loop and input/output values are recorded on an RB. If the command sequence decoded is a loop and there is a complete match between the contents recorded on the RF and the RB about the loop, the output value recorded on the RB is output as an output about the loop.



LEGAL STATUS

[Date of request for examination] 10.02.2006

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

BEST AVAILABLE COPY

Japanese Laid-Open Publication

No. 2004-258905 (*Tokukai 2004-258905*)

A. Relevance of the Above-identified Document

The following is a partial English translation of exemplary portions of non-English language information that may be relevant to the issue of patentability of the claims of the present application.

B. Translation of the Relevant Passages of the Document

See the attached English Abstract.

[Claims]

[Claim 12]

A data processing device defined in any one of claims 9-11,

wherein, in each entry of the instruction row storage means, the state of registration and reuse within a predetermined time is stored, $E = (\text{the number of steps reduced in the past}) \times (\text{the number of registration}) \times (\text{the number of reuse})$ is worked out, and the second computing means subjects, to precomputation, the entry which has the largest E.

(19) 日本国特許庁 (JP)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2004-258905

(P2004-258905A)

(43) 公開日 平成16年9月16日 (2004.9.16)

(51) Int. Cl. ⁷

G06F 9/32

F 1

G06F 9/32 330A

G06F 9/32 330C

テーマコード (参考)

5B033

審査請求 未請求 請求項の数 14 O L (全 31 頁)

(21) 出願番号 特願2003-47909 (P2003-47909)
 (22) 出願日 平成15年2月25日 (2003.2.25)

(71) 出願人 503360115
 独立行政法人 科学技術振興機構
 埼玉県川口市本町4丁目1番8号
 (74) 代理人 100080034
 弁理士 原 謙三
 (74) 代理人 100113701
 弁理士 木島 隆一
 (74) 代理人 100116241
 弁理士 金子 一郎
 (72) 発明者 中島 康彦
 京都府京都市左京区吉田神楽岡町5-28
 -102
 Fターム (参考) 5B033 AA03 CA11 CA13

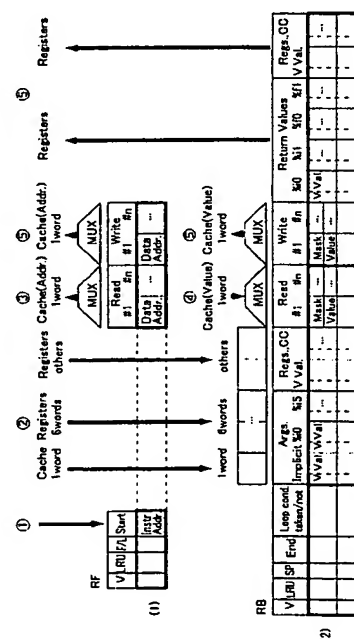
(54) 【発明の名称】 データ処理装置、データ処理プログラム、およびデータ処理プログラムを記録した記録媒体

(57) 【要約】

【課題】 主記憶手段から命令列および／または値を読み出し、演算処理を行った結果を主記憶手段に書き込む処理を行うデータ処理装置において、命令列がループである場合にも、値再利用によって演算の高速化を図ることができるデータ処理装置を提供する。

【解決手段】 R Fに、再利用すべきループの開始アドレス、終了アドレス、および入出力に関する主記憶アドレスを記録し、R Bに、該ループの入出力に関する全てのレジスタの情報および入出力値を記録しておく。そして、デコードした命令列がループであった場合、該ループに関して、R FおよびR Bに登録されている内容と完全に一致した場合、R Bに登録されている出力値を、該ループに関する出力として出力する。

【選択図】 図1



【特許請求の範囲】

【請求項 1】

主記憶手段から命令列および／または値を読み出し、演算処理を行った結果を主記憶手段に書き込む処理を行うデータ処理装置において、
上記主記憶手段から読み出した命令列をデコードし、該命令列に基づく演算を行う第 1 の演算手段と、

1 つ以上の命令列に関する情報を記憶する命令列記憶手段とを備え、

上記第 1 の演算手段が、

命令列をデコードした結果、該命令列が後方分岐命令に挟まれたループであるか否かを判断する第 1 のステップと、

上記第 1 のステップにおいて、ループであると判断された場合に、該ループに関する情報が上記命令列記憶手段に記憶されているか否かを判断する第 2 のステップと、

上記第 2 のステップにおいて、記憶されていないと判断された場合に、該ループを実行する演算を行うとともに、上記命令列記憶手段に対して、演算結果に基づいて該ループに関する入力および出力を記録する第 3 のステップと、

上記第 2 のステップにおいて、記憶されていると判断された場合に、処理対象としてのループに対する入力と、上記命令列記憶手段に記憶されているループに対する入力とが一致している場合に、上記命令列記憶手段に記憶されているループに対する出力の値を、処理対象としてのループに対する出力として出力する第 4 のステップとを行うことを特徴とするデータ処理装置。

【請求項 2】

上記第 1 のステップにおいて、さらに、命令列をデコードした結果、該命令列が関数であるか否かも判断し、

上記第 1 のステップにおいて、関数であると判断された場合に、該関数に関する情報が上記命令列記憶手段に記憶されているか否かを判断する第 5 のステップと、

上記第 5 のステップにおいて、記憶されていないと判断された場合に、該関数を実行する演算を行うとともに、上記命令列記憶手段に対して、演算結果に基づいて該関数に関する入力および出力を記録する第 6 のステップと、

上記第 5 のステップにおいて、記憶されていると判断された場合に、処理対象としての関数に対する入力と、上記命令列記憶手段に記憶されている関数に対する入力とが一致している場合に、上記命令列記憶手段に記憶されている関数に対する出力の値を、処理対象としての関数に対する出力として出力する第 7 のステップとを行うことを特徴とする請求項 1 記載のデータ処理装置。

【請求項 3】

上記第 3 のステップにおいて、上記命令列記憶手段に対して、主記憶手段における該ループの開始アドレスおよび終了アドレスと、全ての入出力に関する主記憶アドレスとを記録するとともに、

上記第 4 のステップにおいて、処理対象としてのループの開始アドレス、ならびに、入力に関する主記憶アドレスが、命令列記憶手段に記憶されているものと完全に一致した場合に、上記命令列記憶手段に記憶されているループに対する出力の値を、処理対象としてのループに対する出力として出力することを特徴とする請求項 1 または 2 記載のデータ処理装置。

【請求項 4】

上記第 6 のステップにおいて、上記命令列記憶手段に対して、主記憶手段における該関数の先頭アドレスと、入出力に関する主記憶アドレスとを記録するとともに、

上記第 7 のステップにおいて、処理対象としての関数の先頭アドレス、ならびに、入力に関する主記憶アドレスが、命令列記憶手段に記憶されているものと完全に一致した場合に、上記命令列記憶手段に記憶されている関数に対する出力の値を、処理対象としての関数に対する出力として出力することを特徴とする請求項 2 記載のデータ処理装置。

【請求項 5】

上記主記憶手段から読み出された入力を一時的に格納し、上記第1の演算手段にわたすとともに、上記第1の演算手段からの出力を一時的に格納し、上記主記憶手段にわたすレジスタをさらに備えるとともに、

上記命令列記憶手段に、ループの入力および出力が格納される全てのレジスタの情報が記録されることを特徴とする請求項3記載のデータ処理装置。

【請求項6】

特定の上位の命令区間が、その内部に1つ以上の下位の命令区間を含んでいる場合に、下位の命令区間の実行中に、該下位の命令区間が行った入出力が上位の命令区間の局所変数である場合には、該下位の命令区間の入出力として上記命令列記憶手段に登録し、該下位の命令区間が行った入出力が上位の命令区間の局所変数でない場合には、該下位の命令区間および上位の命令区間の入出力として上記命令列記憶手段に登録することを特徴とする請求項1ないし5のいずれか一項に記載のデータ処理装置。

10

【請求項7】

上記命令列記憶手段に対して、上位の命令区間および1つ以上の下位の命令区間が登録されている場合に、登録されている命令区間の包含関係の順番をスタック構造で記憶する再利用記憶手段をさらに備えていることを特徴とする請求項6記載のデータ処理装置。

【請求項8】

上記再利用記憶手段に登録可能な命令区間の多重度を超過して、命令区間を検出した場合、その時点で最も上位の命令区間の登録を中止することを特徴とする請求項7記載のデータ処理装置。

20

【請求項9】

少なくとも1つの第2の演算手段をさらに備え、

上記第2の演算手段が、上記第1の演算手段によって処理が行われている命令列に関して、今後入力が予想される予測入力値に基づいて該命令列の演算を行い、その結果を上記命令列記憶手段に対して登録することを特徴とする請求項1ないし8のいずれか一項に記載のデータ処理装置。

【請求項10】

上記第1の演算手段に対して、主記憶手段に対して読み出しおよび書き込みが可能なキャッシュメモリが設けられているとともに、上記第2の演算手段に対して、主記憶手段に対して読み出しのみ可能なキャッシュメモリが設けられており、

30

上記第2の演算手段が、命令列記憶手段に対する入出力記録対象となる主記憶参照には命令列記憶手段に登録されている入出力記録そのものを用い、その他の局所的な参照には、上記第2の演算手段に設けられたローカルメモリを用いるとともに、

上記第1の演算手段が主記憶手段へ書き込みを行う場合には、対応する第2の演算手段のキャッシュラインが無効化されることを特徴とする請求項9記載のデータ処理装置。

【請求項11】

上記命令列記憶手段において、上記第1の演算手段による書き込み範囲と、上記第2の演算手段による書き込み範囲とを分割するとともに、上記第1の演算手段による書き込み範囲では、LRU (least recently used) によってエントリの入れ替えが行われるようにし、上記第2の演算手段による書き込み範囲では、FIFO (First In First Out) によってエントリの入れ替えが行われるようにすることを特徴とする請求項9または10記載のデータ処理装置。

40

【請求項12】

命令列記憶手段における各エントリごとに、一定期間における登録および再利用の状況が記録され、 $E = (\text{過去の削減ステップ数}) \times (\text{登録回数}) \times (\text{再利用回数})$ が算出されるとともに、上記第2の演算手段が、上記Eが最大となるエントリに関して事前実行を行うことを特徴とする請求項9ないし11のいずれか一項に記載のデータ処理装置。

【請求項13】

請求項1ないし12のいずれか一項に記載のデータ処理装置が備える第1の演算手段が行う処理をコンピュータに実行させることを特徴とするデータ処理プログラム。

50

【請求項 1 4】

請求項 1 ないし 1 2 のいずれか一項に記載のデータ処理装置が備える第 1 の演算手段が行う処理をコンピュータに実行させることを特徴とするデータ処理プログラムを記録した記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、主記憶手段から命令列および／または値を読み出し、演算処理を行った結果を主記憶手段に書き込む処理を行うデータ処理装置に関するものである。

【0002】

【従来の技術】

従来、CPU (Central Processing Unit) を始めとするマイクロプロセッサにおいて、演算速度の高速化技術に関する研究開発が盛んに行われている。高速化技術としては、例えばパイプライン、スーパースケーラ、アウトオブオーダー実行、および、レジスタリネーミングなどが挙げられる。

【0003】

パイプラインは、命令の実行処理を数段階に分解し、複数の命令を流れ作業的に同時処理を行う技術である。スーパースケーラは、命令の実行回路を 2 組以上用意し、複数の命令を同時に並行して実行する技術である。アウトオブオーダー実行は、命令の記述順序を無視して、いくつかの連続する命令の中から先に実行可能なものを探して先行処理を行う技術である。レジスタリネーミングは、例えば CISC (Complex Instruction Set Computer) タイプのプロセッサにおいて、従来のプロセッサにおける命令の互換性を保ちながら、汎用レジスタの数を増やすことによって並行処理が行われる確率を増大させる技術である。

【0004】

このように、マイクロプロセッサにおける演算速度の高速化を図る際には、命令の実行を並行して行うことが重要となっている。しかしながら、プログラム中には、ある命令の結果に応じて異なる命令が行われるような依存関係、言い換えれば分岐が含まれている場合がほとんどである。このような分岐が含まれている場合、並行処理によって先行して処理を行っている、分岐の結果によって先行処理した内容が無駄になるという状況が発生することになり、演算速度の高速化の効果が小さくなるという問題がある。

【0005】

そこで、プログラム中に分岐がある場合に、分岐先を予測することによって先行処理が無駄になる確率を低減し、並行処理の効果を向上させる技術、いわゆる分岐予測に関する研究が数多く行われている。

【0006】

しかしながら、分岐予測に基づいて投機的先行処理を行う場合には、一般的に次のような問題がある。第 1 の問題としては、予測の正当性を常に検証する必要があるので、先行命令列の実行時間そのものを削減することはできない、という点である。第 2 の問題としては、誤った予測に基づく一連の先行演算結果を全て無効化する必要があるため、一度に投機的先行処理できる命令数を多くするには、相応のハードウェアコストを要する、という点である。第 3 の問題としては、命令間の依存関係が多いほど、多重に投機的先行処理をする必要が生じ、予測の正当性の検証処理、および誤った予測に基づく処理の無効化処理が極めて複雑になる、という点である。

【0007】

一方、分岐予測とは異なる高速化技術として、値再利用という技術も提案されている。この値再利用とは、プログラムの一部分に関する入力値および出力値を再利用表に登録しておき、同じ箇所を再度実行する際に、入力値が再利用表に登録されているものである場合には、登録されている出力値を出力する、という技術である。この値再利用による効果としては次のようなものが挙げられる。(1) 入力値が、再利用表に登録されている入力値

10

20

30

40

50

と一致すれば、実行結果を検証する必要がない。(2) 入力値および出力値の総数によってのみハードウェアコストが決定され、省略可能な命令列の長さが制約されない。(3) 命令間の依存関係の多少は、再利用機構の複雑さに影響を与えない。(4) 冗長なロード／ストア命令を削減することができるとともに、これに伴う消費電力の削減も実現される。

【0008】

後記する非特許文献1には、プログラムにおける関数に関して値再利用を行う技術が示されている。この従来技術では、一般的にロードモジュールがABI (Application Binary Interface) に従って作られることを利用しており、特に、SPARC (Scalable Processor Architecture)

10

ABIを利用している。そして、このABIにおいて関数の入出力を特定することによって値再利用を実現している。すなわち、値再利用のためのコンパイラによる専用命令の埋め込みが不要となっており、既存ロードモジュールへの適用が可能となっている。

【0009】

また、関数の多重構造を動的に把握することにより、関数内局所レジスタやスタック上の局所変数を値再利用における入出力値から除外するようにしており、これによって効率を向上させている。特に関数については、関数の複雑さに拘わらず、最大6のレジスタ入力、最大4のレジスタ出力、および、局所変数を含まない最小限の主記憶値の登録による再利用および事前実行が可能となっている。この従来技術について以下に詳細に説明する。

【0010】

20

まず、単一の関数を対象として、何が入力で何が出力であるかを明らかにし、1レベルの再利用を行うために必要な機構について説明する。プログラムにおいては、一般的に関数は多重構造を形成している。関数A (Function-A) が関数B (Function-B) を呼び出す構造を図13 (a) に示す。

【0011】

帯域変数 (Globals) は、関数Aの入出力 (A_{in} / A_{out}) および関数Bの入出力 (B_{in} / B_{out}) になりうるものである。関数Aの局所変数 (Locals-A) は、関数Aの入出力ではないが、ポインタを通じてBの入出力になりうるものである。また、関数Aから関数Bへの引数 (Args) は、関数Bへの入力となりうるものであり、関数Bから関数Aの戻り値 (Ret. Val.) は、関数Bからの出力となりうるものである。なお、関数Bの局所変数 (Locals-B) は、関数Aおよび関数Bの入出力には含まれない。

30

【0012】

コンテキストに依存せずに関数Bを再利用するには、関数Bの実行時に、関数Bの入出力 B_{in} / B_{out} のみを入出力として登録しなければならない。ここで、図13 (a) に示すプログラム構造を実行する際の主記憶におけるメモリマップを図13 (b) に示す。このメモリマップにおいて、 B_{in} / B_{out} を含まない領域はLocals-Bのみとなっている。よって、 B_{in} / B_{out} を識別するには、GlobalsとLocals-Bとの境界、および、Locals-BとLocals-Aとの境界をそれぞれ確定しなければならない。前者については、一般的にOS (Operating System) が実行時のデータサイズおよびスタックサイズの上限を決めることを利用し、OSが設定する境界 (LIMIT) に基づいてGlobalsとLocals-Bとの境界を確定することができる。後者については、Bが呼び出される直前のスタックポインタの値 (SP in A) を用いることによって、Locals-BとLocals-Aとの境界を確定することができる。

40

【0013】

次に、与えられた主記憶アドレスが、大域変数であるか、または、どの関数の局所変数であるかを識別する方法について説明する。ロードモジュールは、SPARC ABIに規定されている以下の条件を満たすと仮定する。なお、%fpはフレームポインタ、%spはスタックポインタを意味するものとする。

50

▲1▼%s p 以上の領域のうち、%s p + 0 ~ 6 3 はレジスタ退避領域、%s p + 6 8 ~ 9 1 は引数退避領域であり、いずれも関数の入出力ではない。

▲2▼構造体を返す場合の暗黙的引数 (Implicit Arg.) は %s p + 6 4 ~ 6 7 に格納される。

▲3▼明示的引数 (Explicit Arg.) はレジスタ %o 0 ~ 5、%s p + 9 2 以上の領域に置かれる。

【0014】

まず、大域変数と局所変数とを区別するために、一般的に、OS が実行時のデータサイズおよびスタックサイズの上限を決めることを利用し、次の事項を仮定する。

▲1▼大域変数は L I M I T 未満の領域に置かれる。

▲2▼%s p は、L I M I T 以下になることはなく、L I M I T ~ %s p の領域は無効である。

【0015】

以上の条件を満たしながら、関数 A が関数 B を呼び出す場合の、メモリマップにおける引数およびフレームの概要を図 1 4 に示す。同図を参照しながら、以下に A の局所変数および B の局所変数を区別する方法について説明する。

【0016】

同図において、(a) は A 実行中の状態を示している。L I M I T 未満の太枠部分に命令 (Instructions) および大域変数 (Global Vars.) が格納され、%s p 以上に有効な値が格納されている。%s p + 6 4 には、B が構造体を返り値とする場合の暗黙的引数として、構造体の先頭アドレスが格納される。B に対する明示的引数の先頭 6 ワードはレジスタ %o 0 ~ 5、第 7 ワード以降は %s p + 9 2 以上に格納される。ベースレジスタを %s p とするオペランド %s p + 9 2 が出現した場合、この領域は引数の第 7 ワードすなわち B の局所変数である。一方、オペランド %s p + 9 2 が出現しない場合、この領域は A の局所変数である。このように、(a) の状態では、オペランドを検証することによって A の局所変数と B 局所変数とを区別することができる。

【0017】

一方、(b) は B 実行中の状態を示している。引数が入力、返り値が出力、大域変数および A の局所変数が入出力となりうる。ただし、B は可変長引数を受け入れる場合があるので、一般に %f p + 9 2 以上の領域が A の局所変数の領域となるか B の局所変数の領域となるかは判断できない。

【0018】

局所変数を区別するには、まず、(a) の時点において引数の第 7 ワード以降を検出した関数呼び出しは再利用の対象外とし、第 7 ワード以降を検出しない関数呼び出しに関して、直前に %s p + 9 2 の値を記録しておくようにする。なお、第 7 ワード以降を使用する関数呼び出しの出現頻度が低いと予想されることから、第 7 ワード以降を使用する関数を再利用の対象外とする制限による性能低下は軽微なものとする。

【0019】

以上の準備により、(b) における主記憶参照アドレスが、予め記録した %s p + 9 2 の値以上の場合は A の局所変数、小さい場合は B の局所変数であることがわかる。B 実行時には、B の局所変数を除外しながら、大域変数および A の局所変数を再利用表へ登録する。

【0020】

再利用の際は、B の局所変数は入出力から除外されるので、B の局所変数のアドレスが一致している必要がない。このため、いかなるコンテキストであっても、入力さえ一致すれば、再利用することが可能である。ただし、B が参照する大域変数や A の局所変数については、アドレスおよびデータの両方が再利用表の内容と完全に一致する必要がある。すなわち、B を実行する前に、どのようにして比較すべき主記憶アドレスを網羅するかがポイントになる。

【0021】

10

20

30

40

50

Bが参照する大域変数やAの局所変数のアドレスは、そもそもBにおいて生成されるアドレス定数や、大域変数／引数を起源とするポインタに基づいているものである。よって、まず引数が完全に一致する再利用表中のエントリを選択した後に、関連する主記憶アドレスをすべて参照して一致比較を行うことにより、Bが参照すべき主記憶アドレスを網羅することができる。そして、全ての入力が一一致した場合にのみ、登録済の出力（返り値、大域変数、およびAの局所変数）を再利用することができる。

【0022】

関数再利用を実現するために、再利用表として、関数管理表（RF）および入出力記録表（RB）を設けることにする。1つの関数を再利用するために必要なハードウェア構成を図15に示す。複数の関数を再利用可能とするには、この構成を複数組用意することになる。

10

【0023】

この表において、RFおよびRBに保持されるVは、エントリが有効であるか否かを示すフラグであり、LRU（least recently used）は、エントリ入れ替えのヒントを示している。RFは、上記のVおよびLRUの他に、関数の先頭アドレス（Start）、および参照すべき主記憶アドレス（Read/Write）を保持する。RBは、上記のVおよびLRUの他に、関数呼び出し直前の%sp（SP）、引数（Args.）（V：有効エントリ、Val.：値）、主記憶値（Mask：Read/Writeアドレスの有効バイト、Value：値）、および、返り値（Return Values）（V：有効エントリ、Val.：値）を保持する。

20

【0024】

返り値は、%i0～1（リーフ関数では%o0～1に読み替える）または%f0～1に格納され、%f2～3を使用する返り値（拡張倍精度浮動小数点数）は対象プログラムには存在しないものと仮定する。ReadアドレスはRFが一括管理し、MaskおよびValueはRBが管理することにより、Readアドレスの内容とRBの複数エントリをCAM（content-addressable memory）により一度に比較する構成を可能としている。

【0025】

単一の関数を再利用するには、まず、関数実行時に、局所変数を除外しながら、引数、返り値、大域変数および上位関数の局所変数に関する入出力情報を再利用表に登録していく。ここで、読み出しが先行した引数レジスタは関数の入出力として、また、返り値レジスタへの書き込みは関数の出力として登録する。その他のレジスタ参照は登録する必要がない。主記憶参照も同様に、読み出しが先行したアドレスについては入力、書き込みは出力として登録する。

30

【0026】

関数から復帰するまでに次の関数を呼び出した場合、または、登録すべき入出力が再利用表の容量を超える、引数の第7ワードを検出する、途中でシステムコールや割り込みが発生する、などの擾乱が発生しなかった場合、復帰命令を実行した時点で、登録中の入出力表エントリを有効にする。

【0027】

以降、図15を参照しながら説明すると、関数を呼び出す前に、▲1▼関数先頭アドレスを検索し、▲2▼引数が完全に一致するエントリを選択し、▲3▼関連する主記憶アドレスすなわち少なくとも1つのMaskが有効であるReadアドレスをすべて参照して、▲4▼一致比較を行う。全ての入力が一一致した場合に、▲5▼登録済の出力（返り値、大域変数、およびAの局所変数）を書き戻すことによって、関数の実行を省略することができる。

40

【0028】

【非特許文献1】

情報処理学会論文誌：ハイパフォーマンスコンピューティングシステム，HPS5，pp. 1-12，Sep.（2002），“関数値再利用および並列事前実行による高速化技

50

術”（中島康彦、緒方勝也、正西申悟、五島正裕、森真一郎、北村俊明、富田眞治）（発行日2002年9月15日）

【0029】

【発明が解決しようとする課題】

上記従来の技術は、関数のみを再利用の対象としたものとなっている。よって、コンパイラが関数をインライン展開したり、関数内に処理量の多いループが存在したりする場合には、演算の高速化をほとんど期待することができないことになる。

【0030】

本発明は上記の問題点を解決するためになされたもので、その目的は、主記憶手段から命令列および／または値を読み出し、演算処理を行った結果を主記憶手段に書き込む処理を行うデータ処理装置において、命令列がループである場合にも、値再利用によって演算の高速化を図ることができるデータ処理装置を提供することにある。

【0031】

【課題を解決するための手段】

上記の課題を解決するために、本発明に係るデータ処理装置は、主記憶手段から命令列および／または値を読み出し、演算処理を行った結果を主記憶手段に書き込む処理を行うデータ処理装置において、上記主記憶手段から読み出した命令列をデコードし、該命令列に基づく演算を行う第1の演算手段と、1つ以上の命令列に関する情報を記憶する命令列記憶手段とを備え、上記第1の演算手段が、命令列をデコードした結果、該命令列が後方分岐命令に挟まれたループであるか否かを判断する第1のステップと、上記第1のステップにおいて、ループであると判断された場合に、該ループに関する情報が上記命令列記憶手段に記憶されているか否かを判断する第2のステップと、上記第2のステップにおいて、記憶されていないと判断された場合に、該ループを実行する演算を行うとともに、上記命令列記憶手段に対して、演算結果に基づいて該ループに関する入力および出力を記録する第3のステップと、上記第2のステップにおいて、記憶されていると判断された場合に、処理対象としてのループに対する入力と、上記命令列記憶手段に記憶されているループに対する入力とが一致している場合に、上記命令列記憶手段に記憶されているループに対する出力の値を、処理対象としてのループに対する出力として出力する第4のステップとを行うことを特徴としている。

【0032】

上記の構成では、主記憶から読み出された命令列が第1の演算手段によって演算され、演算結果が主記憶に書き込まれるようになっている。ここで、第1の演算手段は、命令列をデコードした際に、まず該命令列がループであるか否かを判定する。そして、ループである場合には、そのループが初めて実行される場合には、そのループに基づく演算を行うとともに、命令列記憶手段に入力および出力を記録する処理が行われる。その後、同じループが出現した場合に、命令列記憶手段に記憶されている入力と同じであれば、第1の演算手段がそのループに関する演算処理を行うことなしに、命令列記憶手段に記憶されている出力をそのまま出力するようになっている。

【0033】

したがって、あるループに関する命令列が複数回行われる場合には、2回目以降では、第1の演算手段による演算処理を行うことなく、命令列記憶手段に記憶されている出力を読み出すという処理のみで対応することが可能となる。これにより、処理時間の大幅な短縮を実現することができる。

【0034】

また、本発明に係るデータ処理装置は、上記の構成において、上記第1のステップにおいて、さらに、命令列をデコードした結果、該命令列が関数であるか否かも判断し、上記第1のステップにおいて、関数であると判断された場合に、該関数に関する情報が上記命令列記憶手段に記憶されているか否かを判断する第5のステップと、上記第5のステップにおいて、記憶されていないと判断された場合に、該関数を実行する演算を行うとともに、上記命令列記憶手段に対して、演算結果に基づいて該関数に関する入力および出力を記録

する第6のステップと、上記第5のステップにおいて、記憶されていると判断された場合に、処理対象としての関数に対する入力と、上記命令列記憶手段に記憶されている関数に対する入力とが一致している場合に、上記命令列記憶手段に記憶されている関数に対する出力の値を、処理対象としての関数に対する出力として出力する第7のステップとを行う構成としてもよい。

【0035】

上記の構成によれば、命令列が関数である場合にも、上記のループと同様に値再利用を実現することができる。すなわち、ある関数に関する命令列が複数回行われる場合には、2回目以降では、第1の演算手段による演算処理を行うことなく、命令列記憶手段に記憶されている出力を読み出すという処理のみで対応することが可能となる。これにより、処理時間の大幅な短縮を実現することができる。

10

【0036】

また、本発明に係るデータ処理装置は、上記の構成において、上記第3のステップにおいて、上記命令列記憶手段に対して、主記憶手段における該ループの開始アドレスおよび終了アドレスと、全ての入出力に関する主記憶アドレスとを記録するとともに、上記第4のステップにおいて、処理対象としてのループの開始アドレス、ならびに、入力に関する主記憶アドレスが、命令列記憶手段に記憶されているものと完全に一致した場合に、上記命令列記憶手段に記憶されているループに対する出力の値を、処理対象としてのループに対する出力として出力する構成としてもよい。

【0037】

上記の構成によれば、命令列記憶手段には、ループの開始アドレスおよび終了アドレスと、全ての入出力に関する主記憶アドレスとが記録されていることになる。よって、これらの情報を用いることにより、的確にループを特定することが可能となり、ループに関する再利用を実現することが可能となる。

20

【0038】

また、本発明に係るデータ処理装置は、上記の構成において、上記第6のステップにおいて、上記命令列記憶手段に対して、主記憶手段における該関数の先頭アドレスと、入出力に関する主記憶アドレスとを記録するとともに、上記第7のステップにおいて、処理対象としての関数の先頭アドレス、ならびに、入力に関する主記憶アドレスが、命令列記憶手段に記憶されているものと完全に一致した場合に、上記命令列記憶手段に記憶されている関数に対する出力の値を、処理対象としての関数に対する出力として出力する構成としてもよい。

30

【0039】

上記の構成によれば、命令列記憶手段には、関数の先頭アドレスと、入出力に関する主記憶アドレスとが記録されていることになる。よって、これらの情報を用いることにより、的確に関数を特定することが可能となり、関数に関する再利用を実現することが可能となる。

【0040】

また、本発明に係るデータ処理装置は、上記の構成において、上記主記憶手段から読み出された入力を一時的に格納し、上記第1の演算手段にわたすとともに、上記第1の演算手段からの出力を一時的に格納し、上記主記憶手段にわたすレジスタをさらに備えるとともに、上記命令列記憶手段に、ループの入力および出力が格納される全てのレジスタの情報が記録される構成としてもよい。

40

【0041】

上記の構成によれば、第1の演算手段は、レジスタの値に基づいて、再利用が可能か否かの判断を行うことができるので、この判断を的確かつ迅速に行うことが可能となる。

【0042】

また、本発明に係るデータ処理装置は、上記の構成において、特定の上位の命令区間が、その内部に1つ以上の下位の命令区間を含んでいる場合に、下位の命令区間の実行中に、該下位の命令区間が行った入出力が上位の命令区間の局所変数である場合には、該下位の

50

命令区間の入出力として上記命令列記憶手段に登録し、該下位の命令区間が行った入出力が上位の命令区間の局所変数でない場合には、該下位の命令区間および上位の命令区間の入出力として上記命令列記憶手段に登録する構成としてもよい。

【0043】

上記の構成によれば、ある命令区間の内部に、さらに他の命令区間が含まれているような多重入出力構造である場合にも、命令列記憶手段に対して、的確に再利用のための情報を記録することが可能となる。したがって、例えば命令列記憶手段に上位の命令区間に関する情報が登録されている場合に、該命令区間が出現した場合に、その入力を参照することによって、下位の命令区間を考慮することなく、的確に該命令区間の再利用を実現することが可能となる。

10

【0044】

また、本発明に係るデータ処理装置は、上記の構成において、上記命令列記憶手段に対して、上位の命令区間および1つ以上の下位の命令区間が登録されている場合に、登録されている命令区間の包含関係の順番をスタック構造で記憶する再利用記憶手段をさらに備えている構成としてもよい。

【0045】

上記の構成によれば、命令列記憶手段に多重入出力構造の命令区間が複数登録されていても、再利用記憶手段を参照することによって、各命令区間の包含関係を認識することが可能となる。

【0046】

また、本発明に係るデータ処理装置は、上記の構成において、上記再利用記憶手段に登録可能な命令区間の多重度を超えて、命令区間を検出した場合、その時点で最も上位の命令区間の登録を中止する構成としてもよい。

20

【0047】

上記の構成によれば、多重入出力構造を登録中に、その構造に含まれる新たな命令区間を検出し、かつ、再利用記憶手段に登録可能な多重度を超えてしまう場合に、最も上位の命令区間に関する登録が中止されることになる。ここで、例えば最も下位の命令区間に関する登録が中止されてしまうと、それまでに登録されていた多重入出力構造は全て使えない状態になってしまうが、上記のようにすれば、再利用可能な多重入出力構造の登録を最低限確保することが可能となる。

30

【0048】

また、本発明に係るデータ処理装置は、上記の構成において、少なくとも1つの第2の演算手段をさらに備え、上記第2の演算手段が、上記第1の演算手段によって処理が行われている命令列に関して、今後入力が予想される予測入力値に基づいて該命令列の演算を行い、その結果を上記命令列記憶手段に対して登録する構成としてもよい。

【0049】

上記の構成によれば、第2の演算手段によって、その時点で第1の演算手段によって処理が行われている命令列に関して、予測入力値に基づく演算が行われ、その結果が命令列記憶手段に記憶されることになる。よって、次に、同じ命令列が出現し、予測入力値と同じ入力が行われた場合には、命令列記憶手段に記憶されている値を再利用することが可能となる。例えば、入力値が単調に変化するような命令列の場合には、予測入力値が的中する可能性が高いので、上記の構成による効果は高くなる。

40

【0050】

また、本発明に係るデータ処理装置は、上記の構成において、上記第1の演算手段に対して、主記憶手段に対して読み出しおよび書き込みが可能なキャッシュメモリが設けられているとともに、上記第2の演算手段に対して、主記憶手段に対して読み出しのみ可能なキャッシュメモリが設けられており、上記第2の演算手段が、命令列記憶手段に対する入出力記録対象となる主記憶参照には命令列記憶手段に記録されている入出力記録そのものを用い、その他の局所的な参照には、上記第2の演算手段に設けられたローカルメモリを用いるとともに、上記第1の演算手段が主記憶手段へ書き込みを行う場合には、対応する第

50

2の演算手段のキャッシュラインが無効化される構成としてもよい。

【0051】

上記の構成によれば、第2の演算手段は、主記憶手段に対しては読み出しのみを行い、書き込みはできないようになっている。そして、第2の演算手段は、命令列記憶手段に対する入出力記録対象となる主記憶参照には命令列記憶手段に記録されている入出力記録そのものを用い、その他の局所的な参照には、上記第2の演算手段に設けられたローカルメモリを用いるようになっている。これにより、第1の演算手段と、第2の演算手段とで、主記憶手段に対して異なる値を書き込むことによる不具合を防止することができる。

【0052】

また、第1の演算手段が主記憶手段へ書き込みを行う場合には、対応する第2の演算手段のキャッシュラインが無効化されるので、第1の演算手段によって主記憶手段が書き換えられた場合にも、第2の演算手段が、新たに主記憶手段から値を読み出すことによって、主記憶一貫性を保つことができる。

【0053】

また、本発明に係るデータ処理装置は、上記の構成において、上記命令列記憶手段において、上記第1の演算手段による書き込み範囲と、上記第2の演算手段による書き込み範囲とを分割するとともに、上記第1の演算手段による書き込み範囲では、LRU (least recently used) によってエントリの入れ替えが行われるようにし、上記第2の演算手段による書き込み範囲では、FIFO (First In First Out) によってエントリの入れ替えが行われるようにする構成としてもよい。

【0054】

命令区間は、大きく分けると、第1の演算手段のみでも再利用の効果があるものと、配列を扱うループのように第1の演算手段のみでは効果がないものとに分けられると考えられる。前者であれば、LRUによる入れ替え、後者であれば、FIFOによる入れ替えが有効である。

【0055】

これに対して、上記の構成によれば、第1の演算手段による書き込み範囲と、第2の演算手段による書き込み範囲とを分割するとともに、エントリの入れ替えは、第1の演算手段による書き込み範囲ではLRU、第2の演算手段による書き込み範囲ではFIFOとしているので、より効率的に再利用を行うことが可能となる。

【0056】

また、本発明に係るデータ処理装置は、上記の構成において、命令列記憶手段における各エントリごとに、一定期間における登録および再利用の状況が記録され、 $E = (\text{過去の削減ステップ数}) \times (\text{登録回数}) \times (\text{再利用回数})$ が算出されるとともに、上記第2の演算手段が、上記Eが最大となるエントリに関して事前実行を行う構成としてもよい。

【0057】

上記の構成によれば、第1の演算手段による登録頻度が高く、かつ、第2の演算手段が登録したエントリの再利用頻度も高い命令列を継続して第2の演算手段の実行対象とすることができる。よって、第2の演算手段による事前実行の効果を最大限高めることが可能となる。

【0058】

【発明の実施の形態】

本発明の実施の一形態について図1ないし図12に基づいて説明すれば、以下のとおりである。

【0059】

(データ処理装置の構成)

本実施形態に係るデータ処理装置の概略構成を図5に示す。同図に示すように、該データ処理装置は、MSP (Main Stream Processor) 1A、SSP (Shadow Stream Processor) 1B、再利用表としてのRF/RB (命令列記憶手段) 2、および主記憶 (主記憶手段) を備えた構成となっており、主記憶3

10

20

30

40

50

に記憶されているプログラムデータなどを読み出して各種演算処理を行い、演算結果を主記憶3に書き込む処理を行うものである。なお、同図に示す構成では、SSP1Bを1つ備えた構成となっているが、2つ以上備えた構成となってもよい。また、同図に示す構成では、SSP1Bを備えた構成となっているが、SSP1Bを備えていない構成としてもかまわない。SSP1Bを備えた場合の作用・効果については、後述する。

【0060】

RF/RB2は、プログラムにおける関数およびループを再利用するためのデータを格納するメモリ手段である。このRF/RB2の詳細については後述する。

【0061】

主記憶3は、MSP1AおよびSSP1Bの作業領域としてのメモリであり、例えばRAM(Random Access Memory)などによって構成されるものである。例えばハードディスクなどの外部記憶手段からプログラムやデータなどが主記憶3に読み出され、MSP1AおよびSSP1Bは、主記憶3に読み出されたデータに基づいて演算を行うことになる。

【0062】

MSP1Aは、RW(再利用記憶手段)4A、演算器(第1の演算手段)5A、レジスタ6A、およびCache7Aを備えた構成となっている。また、SSP1Bは、同様に、RW(再利用記憶手段)4B、演算器(第2の演算手段)5B、レジスタ6B、およびCache/Local7Bを備えた構成となっている。

【0063】

RW4A・4Bは、再利用ウィンドウであり、現在実行中かつ登録中であるRFおよびRB(後述する)の各エントリをリング構造のスタックとして保持するものである。このRW4A・4Bは、実際のハードウェア構造としては、RF/RB2における特定のエントリをアクティブにする制御線の集合によって構成される。

【0064】

演算器5A・5Bは、レジスタ6A・6Bに保持されているデータに基づいて演算処理を行うものであり、ALU(arithmetic and logical unit)と呼ばれるものである。レジスタ6A・6Bは、演算器5A・5Bによって演算を行うためのデータを保持する記憶手段である。なお、本実施形態では、演算器5A・5B、およびレジスタ6A・6Bは、SPARCアーキテクチャに準じたものとする。Cache7A・7Bは、主記憶3と、MSP1AおよびSSP1Bとの間でのキャッシュメモリとして機能するものである。なお、SSP1Bでは、Cache7Bには、局所メモリとしてのLocal7Bが含まれているものとする。

【0065】

(関数およびループの概念的説明)

ここで、上記データ処理装置の詳細な説明に入る前に、関数とループとの類似性について、図2(a)および図2(b)を参照しながら説明しておく。同図(a)は、関数が呼び出されて実行される様子を概念的に示しており、同図(b)は、ループ処理が実行される様子を概念的に示している。関数に含まれる命令は、Call命令の分岐先(Func. start)からReturn命令(Func. end)(厳密にはディレイスロットを含む)までである。同様に、ループでは、後方分岐命令の分岐先(Loop start)から、同じ後方分岐命令(Loop end)までである。したがって、このLoop startからLoop endの間の入出力を登録しておくことによって、関数と同様にループを再利用することができる。ただし、関数では局所変数の登録を除外することができる一方、ループでは除外することができない。これは、ABIでは、ループ内の局所変数を特定することができないからである。したがって、ループを再利用するためには、図15に示した従来の関数再利用表に対して、新たに項目を拡張する必要があることになる。

【0066】

(RF/RBの構成)

10

20

30

40

50

図1は、本実施形態におけるRF/RB2によって実現される再利用表を示している。同図に示すように、RFは、エントリが有効であるか否かを示すV、エントリ入れ替えのヒントを示すLRU、関数の先頭アドレスを示すStart、参照すべき主記憶アドレスを示すRead/Writeに加えて、および、関数とループとを区別するF/Lを保持している。

【0067】

また、RBは、エントリが有効であるか否かを示すV、エントリ入れ替えのヒントを示すLRU、関数またはループを呼び出す際の直前のスタックポイント%spを示すSP、引数(Args.) (V:有効エントリ、Val.:値)、主記憶値(Mask:Read/Writeアドレスの有効バイト、Value:値)、および、返回值(Return Values) (V:有効エントリ、Val.:値)に加えて、ループの終了アドレス(End)、ループ終了時の分岐方向を示すtaken/not、引数や返回值以外のレジスタおよび条件コード(Regs., CC)を保持している。

【0068】

上記のRFおよびRBにおける各項目についてより詳細に説明する。上記Vは、上記のようにエントリが有効であるか否かを示すものであるが、具体的には、未登録時には「0」、登録中である場合には「2」、登録済である場合には「1」の値が格納されるようになっている。例えば、RFまたはRBを確保する際に、未登録エントリ(V=0)があれば、これを使用し、未登録エントリがなければ、登録済エントリ(V=1)の中からLRUが最小のものを選択して上書きすることになる。登録中エントリ(V=2)は使用中であるので上書きすることはできない。

【0069】

上記LRUは、一定時間間隔で右へシフトされていくシフトレジスタの中の「1」の個数を示したものである。RFの場合、このシフトレジスタは、該当エントリに関して、再利用のための登録を行ったか、もしくは再利用を試みた場合に、左端に「1」が書き込まれるようになっている。したがって、該当エントリが頻繁に使用されれば、LRUは大きな値となり、一定期間使用されなければ、LRUの値は0となる。一方、RBの場合、シフトレジスタには、該当エントリが再利用された場合に「1」が書き込まれるようになっている。したがって、該当エントリが頻繁に使用されれば、LRUは大きな値となり、一定期間使用されなければ、LRUの値は0となる。

【0070】

上記RBにおける主記憶値のMaskについて説明する。一般に、アドレスとデータとを1バイトずつ管理することにすれば管理が可能であるが、実際には、4バイト単位でデータを管理の方がキャッシュ参照を高速に行うことができる。そこで、RFでは、主記憶アドレスを4の倍数で記憶するようになっている。一方、管理単位を4バイトとする場合、1バイト分だけをロードすることに対応できるようにするために、4バイトのうちでどのバイトが有効であるかを示す必要がある。すなわち、Maskは、4バイトのうちでどのバイトが有効であるかを示す4ビットのデータとなっている。例えば、C001番地から1バイト分をロードした結果、値がE8であった場合、RFには、アドレスC000が登録され、RBのMaskに「0100」、Valueに「00E80000」が登録されることになる。

【0071】

上記の引数や返回值以外のレジスタおよび条件コード(Regs., CC)について説明する。本実施形態では、SPARCアーキテクチャレジスタのうち、汎用レジスタ%g0-7、%o0-7、%l0-7、%i0-7、浮動小数点レジスタ%f0-31、条件コードレジスタICC、浮動小数点条件コードレジスタFCCを用いるようになっている(詳細は後述する)。このうち、リーフ関数の入力汎用レジスタ%o0-5、出力汎用レジスタ%o0-1、また、非リーフ関数の入力汎用レジスタ%i0-5、出力汎用レジスタ%i0-1、になり、入力は、arg[0-5]、出力は、rti[0-1]に登録される。SPARC-ABIの規定では、これら以外のレジスタは関数の入出力には

10

20

30

40

50

ならないので、関数に関してはR Bにおける引数 (A r g s .) の項で十分である。

【0072】

一方、S P A R C - A B I の規定では、ループの入出力に関しては、用いられるレジスタの種類を特定することはできないので、ループの入出力を特定するには、全ての種類のレジスタに関してR Bに登録する必要がある。よって、R BにおけるR e g s . , C Cには、% g 0 - 7、% o 0 - 7、% l 0 - 7、% i 0 - 7、% f 0 - 3 1、I C C、F C Cが登録されるようになっている。

【0073】

以上のように、R F / R B 2において、R e a dアドレスはR Fが一括管理し、M a s kおよびV a l u eはR Bが管理している。これにより、R e a dアドレスの内容とR Bの複数エントリをC A M (c o n t e n t - a d d r e s s a b l e m e m o r y) によって一度に比較する構成を可能としている。このことについて、以下により詳しく説明する。

【0074】

一般的に、アドレスが与えられると、そのアドレスに格納された値を参照することができるメモリは、R A Mと呼ばれるメモリである。一方、上記のC A Mとは、連想メモリと呼ばれるメモリであり、検索すべき内容が与えられると、そのエントリに対応する信号がO Nとなるように動作するようになっている。通常は、C A MはR A Mとセットにして用いられる。

【0075】

ここで、C A MとR A Mとの連携動作について、具体例を挙げて説明する。C A Mに、「5, 5, 5, 5, 5」、「1, 3, 1, 1, 1」、「1, 3, 3, 5, 2」、「6, 6, 6, 6, 6」というデータ列がエントリとして登録されており、R A Mに、C A Mにおける各データ列に対応して、「5, 5」、「1, 1」、「1, 2」、「6, 6」というデータが登録されているとする。ここで、検索すべきデータ列として、「1, 3, 3, 5, 2」をC A Mに入力すると、一致するエントリがO Nとなり、R A Mに登録されている該当するデータ「1, 2」が出力されることになる。この具体例と同様の構成および動作によって、上記R Bが実現されることになる。

【0076】

(再利用処理の概略)

次に、関数およびループのそれぞれの場合について、再利用処理の概略について説明する。

【0077】

まず、関数の場合について説明する。関数から復帰するまでに次の関数を呼び出した場合、または、登録すべき入出力が再利用表の容量を超える、引数の第7ワードを検出する、途中でシステムコールや割り込みが発生する、などの擾乱が発生しなかった場合、復帰命令を実行した時点で、登録中の入出力表エントリを有効にする。

【0078】

以降、図1を参照しながら説明すると、関数を呼び出す前に、▲1▼R Fに登録されている関数の先頭アドレスに、該当関数の先頭アドレスと一致するものがあるかを検索する。一致するものがある場合には、▲2▼R Bに登録されている該当関数に関するエントリにおける引数が、呼び出す関数の引数と完全に一致するエントリを選択する。そして、▲3▼関連する主記憶アドレスすなわち少なくとも1つのM a s kが有効であるR e a dアドレスをR Fからすべて参照して、▲4▼R Bに登録されている内容と一致比較を行う。全ての入力が一一致した場合に、▲5▼R Bに登録済の出力(返り値、大域変数、およびAの局所変数)を主記憶に書き戻すことによって、関数の実行を省略する、すなわち関数の再利用を実現することができる。

【0079】

次に、ループの場合について説明する。ループが完了する以前に関数から復帰したり、前記した擾乱が発生したりするなど、ループの入出力登録が中止されなければ、登録中のル

10

20

30

40

50

ープに対応する後方分岐命令を検出した時点で、登録中の入出力表エントリを有効にし、そのループの登録を完了する。

【0080】

さらに、後方分岐命令が成立する場合は、次のループが再利用可能かどうかを判断する。すなわち、図1を参照しながら説明すると、後方分岐する前に、▲1▼RFに登録されているループの先頭アドレスに、該当ループの先頭アドレスと一致するものがあるかを検索する。一致するものがある場合には、▲2▼RBに登録されている該当ループに関するレジスタ入力値が、呼び出すループのレジスタ入力値と完全に一致するエントリを選択する。そして、▲3▼関連する主記憶アドレスをRFから全て参照して、▲4▼RBに登録されている内容と一致比較を行う。全ての入力が一一致した場合に、▲5▼RBに登録済の出力（レジスタおよび主記憶出力値）を主記憶に書き戻すことによってループの実行を省略する、すなわちループの再利用を実現することができる。

10

【0081】

再利用した場合、RBに登録されている分岐方向に基づいて、さらに次のループに関して同様の処理を繰り返す。一方、次のループが再利用不可能であれば、次のループを通常に実行し、RFおよびRBへの登録を開始する。

【0082】

（ループを含む多重再利用）

1レベルで上記のような再利用機構を用いた場合、図13(a)に示した例で言えば、リーフ関数としての関数Bや、関数Bの内部にあるループCなどをそれぞれ再利用することが可能となる。これに対して、ある関数を一度実行しただけで、その関数の内部に含まれる関数やループを含む全ての命令区間が再利用可能となるように登録を行う仕組みが多重再利用である。例えば上記の例で言えば、多重再利用によれば、関数Aを一度実行しただけで、入れ子関係にあるA、B、Cの全ての命令区間が再利用可能となる。以下に、多重再利用を実現する上で必要とされる機能拡張について説明する。

20

【0083】

図8に、一例として、関数Aおよび関数Dの概念的な構造を示す。同図に示す例では、関数Aの内部にループBが存在しており、ループBの内部にループCが存在しており、ループCにおいて関数Dが呼び出されるようになっている。そして、関数Dの内部にループEが存在しており、ループEの内部にループFが存在している。

30

【0084】

図3は、図8に示す関数A、DおよびループB、C、E、Fの入れ子構造において、内側の構造のレジスタ入出力（太枠セル領域）が、外側の構造のレジスタ入出力となる影響範囲（矢印）について示している。例えば、ループFの内部において入力として参照された%i0～5は、ループEおよび関数Dに対する入力でもあり、さらに、関数Dを呼び出したループCおよびループBに対する入力（ただし%o0～5に読み替える）でもある。一方、関数Aにとって%o0～5は局所変数に相当するので、%i0～5（%o0～5）は、関数Aに対してのレジスタ入力とはならない。すなわち、%i0～5（%o0～5）の影響範囲はループBまでとなる。別の見方をすれば、関数Dの内部で%i0～5が参照された場合には、ループBが直接的に%o0～5を参照しなくても、%o0～5をループBの入力値として登録する必要がある。ループF内部において出力された%i0～1についても同様である。

40

【0085】

浮動小数点レジスタはレジスタウィンドウに含まれないので、出力された%f0～1は、関数Aを含む全階層の出力となる。一方、その他のレジスタ入出力は、関数を超えて影響がおよぶことはない。すなわち、ループF内部における入出力、すなわち、レジスタ入力としての%i6～7、%g, l, o、%f0～31、%icc、%fcc、およびレジスタ出力としての%I2～7、%g, l, o、%f2～31、%icc、%fccの影響範囲はループEまでとなる。主記憶に対する入出力については、前述した、関数呼び出し直前の%sp（SP）と比較する方法を入れ子の全階層に対して適用することにより、影響

50

範囲を特定することができる。

【0086】

以上のことから、多重再利用を実現するには、前述したR FおよびR Bを関数やループの入れ子構造と関連づける機構が必要である。図4に示すように、再利用ウィンドウ(R W)を装備することによって、現在実行中かつ登録中であるR FおよびR Bの各エントリ(図中ではA、B、Cと示す)をスタック構造として保持する。関数やループの実行中は、R Wに登録されている全てのエントリについて、これまでに述べた方法に基づいて、レジスタおよび主記憶参照を登録していく。

【0087】

この際に、あるエントリに関して、(1)登録可能項目数の超過、(2)引数の第7ワードの検出、(3)システムコールの検出、によって再利用不可能であると判断した場合には、R Wを用いて、そのエントリに対応するR Bおよび上位のR Bを特定し、登録を中止することができる。

10

【0088】

なお、R Wの深さは有限であるものの、一度に登録可能な多重度を超過して関数やループを検出した場合には、外側の命令区間から順次登録を中止し、より内側の命令区間を登録対象に加えることによって、入れ子関係の動的変化に追従することができる。また、実行および登録中(例えばA)に、再利用可能な命令区間(例えばD)に遭遇した場合には、登録済の入出力をそのまま登録中エントリに追加することによって、R Wの深さを超えるAの多重再利用も可能となる。

20

【0089】

(処理の流れ)

次に、命令がデコードされた場合の具体的な処理の流れについて説明する。以下では、命令がデコードされた結果、関数呼び出し命令である場合、関数復帰命令である場合、後方分岐成立の場合、後方分岐不成立の場合、およびその他の命令の場合について、それぞれ処理の流れを説明する。

【0090】

(関数呼び出し命令である場合)

命令がデコードされた結果、関数呼び出し命令である場合の処理を図9に示すフローチャートを参照しながら以下に説明する。まずステップ1(以降、S 1のように称する)において、引数の第7ワードを検出したか否かが判定される。S 1においてYES、すなわち、引数の第7ワードを検出したと判定された場合には、R Wに登録されている登録中R Bエントリを全て無効化し、S 6に移行して、プログラムカウンタを関数の先頭へ進め、処理を終了する。

30

【0091】

一方、S 1においてNO、すなわち、引数の第7ワードを検出していないと判定された場合には、該関数呼び出しおよび入力値がR FおよびR Bに登録されているか否かを検索する(S 2)。S 2においてYES、すなわち、該関数呼び出しおよび入力値がR FおよびR Bに登録されていると判定された場合には、後述するS 7のステップに移行する。

【0092】

S 2においてNO、すなわち、該関数呼び出しおよび入力値がR FおよびR Bに登録されていないと判定された場合、該関数のためのR FエントリおよびR Bエントリを確保しようと試み、▲1▼既存のR Fエントリがあるか、▲2▼登録作業中につき追い出すことのできないR Fエントリ以外に、使用可能なR Fエントリがあるか、または▲3▼登録作業中につき追い出すことのできないR Bエントリ以外に、使用可能なR Bエントリがあるかを判定する(S 3)。

40

【0093】

S 3においてNO、すなわち、使用可能なR F・R Bエントリがないと判定された場合には、登録を開始せず、R Wに登録されているR Bを全て無効化し(S 5)、R Wを空にする。一方、S 3においてYES、すなわち、使用可能なR F・R Bエントリがあると判定

50

された場合には、該関数のための R F エントリおよび R B エントリを確保し、R W に登録する (S 4)。ここで、R W に登録した際に、R W に登録されている R W エントリが溢れた際には、最も古い R W エントリを削除し、対応する R B を無効化する。S 3 または S 4 が行われた後に、プログラムカウンタを関数の先頭へ進め (S 6)、処理を終了する。

【0094】

一方、S 2 において Y E S、すなわち、該関数呼び出しおよび入力値が R F および R B に登録されていると判定された場合、該関数は再利用可能であることになる。すなわち、R B から出力値を求めるとともに、レジスタおよび主記憶にこの出力値を書き込む (S 7)。そして、登録中の関数／ループが R W に登録されているか否かを判定し (S 8)、登録されている場合には、再利用を行った関数の R B エントリの内容のうち必要なものを R W に登録されているエントリに追加する (S 9)。ここで、R W の T O P から順に登録し、途中で R B があふれた場合には、以降、R W の B O T T O M までに対する R B を無効化し、R W から削除する。その後、プログラムカウンタを次の命令へ進め (S 10)、処理を終了する。

【0095】

(関数復帰命令である場合)

命令がデコードされた結果、関数復帰命令である場合の処理を図 10 に示すフローチャートを参照しながら以下に説明する。S 11 において、R W の T O P から順にたどり、関数に対応する R F / R B が検出されるまでに、ループに関する R B が検出されるか否かが判定される (S 12)。ここでループに関する R B が検出されると (S 12 において Y E S)、該当 R B を全て無効化するとともに、R W から削除する (S 13)。

【0096】

一方、R W 探索中に、該関数に対応する R F / R B を検出したか否かが判定される (S 14)。ここで、該関数に対応する R F / R B が検出されると (S 14 において Y E S)、該当 R B エントリを有効化するとともに、R W から削除する (S 15)。

【0097】

その後、復帰命令を実行し (S 16)、処理を終了する。

【0098】

(後方分岐成立である場合)

命令がデコードされた結果、後方分岐成立である場合の処理を図 11 に示すフローチャートを参照しながら以下に説明する。まず、R W の T O P から順にたどり、関数に対応する R B を検出するか否かが判定される (S 21)。S 21 において Y E S、すなわち、関数に対応する R B を検出した場合には、後述する S 24 のステップに移行する。

【0099】

一方、S 21 において N O、すなわち、関数に対応する R B を検出しない場合には、次に、該後方分岐命令自身のアドレスと R B 中のループ終了アドレスとが一致するか否かが判定される (S 22)。S 22 において N O、すなわち、該後方分岐命令自身のアドレスと R B 中のループ終了アドレスとが一致しないと判定されると、後述する S 24 のステップに移行する。

【0100】

S 22 において Y E S、すなわち、該後方分岐命令自身のアドレスと R B 中のループ終了アドレスとが一致すると判定された場合、R W の T O P から該 R B の手前までの R B を全て無効化し (S 23)、R W から削除する。また、該 R B エントリを有効化し、かつ t a k e n = 1 とし、R W から削除する。

【0101】

次に、S 24 において、次ループの先頭アドレスおよび入力値が R F および R B に登録されているか否かが判定される。S 24 において Y E S、すなわち、次ループの先頭アドレスおよび入力値が R F および R B に登録されている場合には、後述する S 30 のステップに移行する。

【0102】

10

20

30

40

50

一方、S 2 4においてN O、すなわち、次ループの先頭アドレスおよび入力値がR FおよびR Bに登録されていない場合には、次ループのためのR FエントリおよびR Bエントリを確保しようと試み、▲ 1 ▼既存のR Fエントリがあるか、▲ 2 ▼登録作業中につき追い出すことができないR Fエントリ以外に、使用可能なR Fエントリがあるか、または▲ 3 ▼登録作業中につき追い出すことができないR Bエントリ以外に、使用可能なR Bエントリがあるかが判定される（S 2 5）。

【0103】

S 2 5においてN O、すなわち、使用可能なR F・R Bエントリがないと判定された場合には、登録を開始せずに、R Wに登録されているR Bを全て無効化し（S 2 6）、R Wを空にする。その後、S 2 9において、プログラムカウンタを条件分岐先へ進め、処理を終了する。

10

【0104】

一方、S 2 5においてY E S、すなわち、使用可能なR F・R Bエントリがあると判定された場合には、その使用可能なR F・R Bエントリを確保し、確保したR F・R BをR Wに登録する（S 2 7）。また、R Bにループ終了アドレス（後方分岐命令自身のアドレス）に登録する。ここで、R Wへの登録を行った際にR Wが溢れた場合には、最も古いR Wエントリを削除し（S 2 8）、それに対応するR Bを無効化する。その後、S 2 9において、プログラムカウンタを条件分岐先へ進め、処理を終了する。

【0105】

一方、前記したS 2 4においてY E Sとなった場合、次ループは再利用可能であることになるので、R Bから出力値を求め、この値をレジスタおよび主記憶に書き込む（S 3 0）。ここで、登録中の関数／ループがR Wに登録されているか否かが判定され（S 3 1）、登録されている場合、再利用を行ったループのR Bエントリの内容のうち必要なものをR Wに登録されているエントリに追加する（S 3 2）。このとき、R WのT O Pから順に登録し、途中でR Bが溢れた場合、以降、R WのB O T T O Mまでに対するR Bを無効化し、R Wから削除する。

20

【0106】

その後、プログラムカウンタは、次ループ先頭ではなく、該R B中のt a k e nの値に応じて、t a k e n = 1の場合は自命令、t a k e n = 0の場合は、R B中に記憶しておいたループ終了アドレスの次へ進める。その後、処理を終了する。

30

【0107】

（後方分岐不成立である場合）

命令がデコードされた結果、後方分岐不成立である場合の処理を図 1.2 に示すフローチャートを参照しながら以下に説明する。まず、R WのT O Pから順に検索し（S 4 1）、関数に対応するR Bを検出したか否かが判定される（S 4 2）。S 4 2においてY E S、すなわち、関数に対応するR Bを検出したと判定された場合、S 4 6においてプログラムカウンタを次命令に進め、処理を終了する。

【0108】

S 4 2においてN O、すなわち、関数に対応するR Bを検出していないと判定された場合、該後方分岐命令自身のアドレスとR B中のループ終了アドレスが一致するか否かが判定される（S 4 3）。S 4 3においてN O、すなわち、該後方分岐命令に対応するR F／R Bを検出していないと判定された場合、S 4 6においてプログラムカウンタを次命令に進め、処理を終了する。

40

【0109】

一方、S 4 3においてY E S、すなわち、該後方分岐命令に対応するR F／R Bを検出したと判定された場合、R WのT O Pから該R Bの手前までのR Bを全て無効化し（S 4 4）、R Wから削除する。また、該R Bエントリを有効化し、かつt a k e n = 0とし、R Wから削除する（S 4 5）。その後、S 4 6においてプログラムカウンタを次命令に進め、処理を終了する。

【0110】

50

(その他の命令である場合)

次に、命令がデコードされた結果、上記以外のその他の命令である場合について説明する。その他の命令である場合、レジスタR/W、主記憶R/Wが実行される。その際に、RWが空でなければ、以下の手順によってレジスタR/W、主記憶R/WをRWに登録されているRBに対して登録する。以下では、(1)汎用レジスタREADの場合、(2)汎用レジスタWRITEの場合、(3)浮動小数点レジスタREADの場合、(4)浮動小数点レジスタWRITEの場合、(5)条件コードレジスタICC-READの場合、(6)条件コードレジスタICC-WRITEの場合、(7)浮動小数点条件コードレジスタFCC-READの場合、(8)浮動小数点条件コードレジスタFCC-WRITEの場合、(9)主記憶READの場合、(10)主記憶WRITEの場合についてそれぞれ説明する。

【0111】

(1) 汎用レジスタREADの場合

まず、RWのTOPからBOTTOMまで順にたどる。そして、(1-1) 該RBがリーフ関数かつ%o0-6の場合、または該RBが非リーフ関数かつ%i0-6の場合、arg[0-5]、V=0であれば、arg[0-5]、V=1に変更し、arg[0-5]、Valに読み出しデータを記録する。その後、さらにRWをたどり、該RBが関数の場合、処理を終了する。一方、該RBが関数ではない(ループである)場合、arg[0-5]、V=0であれば、arg[0-5]、V=1に変更し、arg[0-5]、Valに読み出しデータを記録し、処理を終了する。

【0112】

一方、(1-2) 該RBがループの場合、(a) %g0-7でgrr[0-7]、V=0であれば、grr[0-7]、V=1に変更し、grr[0-7]、Valに読み出しデータを記録し、処理を終了する。(b) %o0-7でarg[0-7]、V=0であれば、arg[0-7]、V=1に変更し、arg[0-7]、Valに読み出しデータを記録し、処理を終了する。(c) %l0-7でlrr[0-7]、V=0であれば、lrr[0-7]、V=1に変更し、lrr[0-7]、Valに読み出しデータを記録し、処理を終了する。(d) %i0-7でirr[0-7]、V=0であれば、irr[0-7]、V=1に変更し、irr[0-7]、Valに読み出しデータを記録し、次のRWエントリに進む。

【0113】

(2) 汎用レジスタWRITEの場合

まず、RWのTOPからBOTTOMまで順にたどる。そして(2-1) 該RBがリーフ関数かつ%o0-5の場合、または該RBが非リーフ関数かつ%i0-5の場合、arg[0-5]、V=0であれば、以降の読み出しは入力ではないことを示すために、arg[0-5]、V=2に変更する。さらに、%o0-1/%i0-1について、rti[0-1]、V=1に変更し、rti[0-1]、Valに書き込みデータを記録する。その後、さらにRWをたどり、該RBが関数の場合、処理を終了する。一方、該RBが関数ではない(ループである)場合、arg[0-1]、V=0であれば、以降の読み出しは入力ではないことを示すために、arg[0-1]、V=2に変更し、rti[0-1]、V=1に変更し、rti[0-1]、Valに書き込みデータを記録し、処理を終了する。

【0114】

一方、(2-2) 該RBがループの場合、(a) %g0-7でgrr[0-7]、V=0であれば、grr[0-7]、V=2に変更し、grr[0-7]、Valに書き込みデータを記録し、処理を終了する。(b) %o0-7でarg[0-7]、V=0であれば、arg[0-7]、V=2に変更し、arg[0-7]、Valに書き込みデータを記録し、処理を終了する。(c) %l0-7でlrr[0-7]、V=0であれば、lrr[0-7]、V=2に変更し、lrr[0-7]、Valに書き込みデータを記録し、処理を終了する。(d) %i0-7でirr[0-7]、V=0であれば、irr[0-7]

]. $V = 2$ に変更し、 $irr[0-7]$. Val に書き込みデータを記録し、次の RW エントリに進む。

【0115】

(3) 浮動小数点レジスタ $READ$ の場合

まず、 RW の TOP から $BOTTOM$ まで順にたどる。そして (3-1) 該 RB が関数の場合、何もせずに処理を終了する。一方、(3-2) 該 RB がループの場合、 $frr[0-31]$. $V = 0$ であれば、 $frr[0-31]$. $V = 1$ に変更し、 $frr[0-31]$. Val に読み出しデータを記録し、処理を終了する。

【0116】

(4) 浮動小数点レジスタ $WRITE$ の場合

まず、 RW の TOP から $BOTTOM$ まで順にたどる。そして (4-1) 該 RB が関数かつ $\%f0-1$ の場合、 $rtf[0-1]$. $V = 1$ に変更し、 $rtf[0-1]$. Val に書き込みデータを記録する。さらに RW をたどり、 $frr[0-1]$. $V = 0$ であれば、以降の読み出しは入力ではないことを示すために、 $frr[0-1]$. $V = 2$ に変更し、 $rtf[0-1]$. $V = 1$ に変更し、 $rtf[0-1]$. Val に書き込みデータを記録し、処理を終了する。

【0117】

一方、(4-2) 該 RB がループの場合、 $frr[0-31]$. $V = 0$ であれば、 $frr[0-31]$. $V = 2$ に変更し、 $frrw[0-31]$. $V = 1$ に変更し、 $frrw[0-7]$. Val に書き込みデータを記録し、処理を終了する。

【0118】

(5) 条件コードレジスタ $ICC-READ$ の場合

まず、 RW の TOP から $BOTTOM$ まで順にたどる。そして (5-1) 該 RB が関数の場合、何もせずに処理を終了する。一方、(5-2) 該 RB がループの場合、 icr . $V = 0$ であれば、 icr . $V = 1$ に変更し、 icr . Val に読み出しデータを記録し、処理を終了する。

【0119】

(6) 条件コードレジスタ $ICC-WRITE$ の場合

まず、 RW の TOP から $BOTTOM$ まで順にたどる。そして (6-1) 該 RB が関数の場合、何もせずに処理を終了する。一方、(6-2) 該 RB がループの場合、 icr . $V = 0$ であれば、 icr . $V = 2$ 、 icw . $V = 1$ に変更し、 icw . Val に書き込みデータを記録し、処理を終了する。

【0120】

(7) 浮動小数点条件コードレジスタ $FCC-READ$ の場合

まず、 RW の TOP から $BOTTOM$ まで順にたどる。そして (7-1) 該 RB が関数の場合、何もせずに処理を終了する。一方、(7-2) 該 RB がループの場合、 fcr . $V = 0$ であれば、 fcr . $V = 1$ に変更し、 fcr . Val に読み出しデータを記録し、処理を終了する。

【0121】

(8) 条件コードレジスタ $ICC-WRITE$ の場合

まず、 RW の TOP から $BOTTOM$ まで順にたどる。そして (8-1) 該 RB が関数の場合、何もせずに処理を終了する。一方、(8-2) 該 RB がループの場合、 fcr . $V = 0$ であれば、 fcr . $V = 2$ 、 fcw . $V = 1$ に変更し、 fcw . Val に書き込みデータを記録し、処理を終了する。

【0122】

(9) 主記憶 $READ$ の場合

まず、 RW の TOP から $BOTTOM$ まで順にたどる。そして、 RB に $WRITE$ データとして登録済である場合は、その値を使用する。一方、上記の場合ではなく、 RB に $READ$ データとして登録済である場合には、その値を使用する。さらに、いずれにも登録済でない場合は、キャッシュを経由して主記憶から読み込む。

10

20

30

40

50

【0123】

その後、再度RWのTOPからBOTTOMまで順にたどる。そして、(a) アドレスが、RBに登録されているsp+64の場合、構造体ポインタの読み出しであるので、arg0.V=0であれば、arg0.V=1に変更し、arg0.Valに読み出しデータを記録する。(b) 上記の(a)の場合でなく、アドレスが、LIMIT以上sp+92未満であれば、登録不要領域であるので、何もしない。(c) 上記の(b)の場合でない場合、WRITEデータとして登録済であるかどうかを検査し、そうであれば、すでに上書きされたあとのREADであるので登録不要であり、何もしない。(d) 上記の(c)でない場合、READデータとして登録済であるかどうかを検査し、そうであれば、すでに登録済であるので登録不要であり、何もしない。(e) 上記の(d)でない場合、READデータとしての登録が必要であるので、RFに主記憶READアドレスを確保し、READデータとして登録する。RFに主記憶アドレスを確保できなかった場合には、登録不能であるため、そのRWエントリからBOTTOMまでに対応するRBエントリを全て無効化する。

10

【0124】

(10) 主記憶WRITEの場合

まず、キャッシュを経由して、主記憶に書き込む。そして、ベースレジスタが14(%sp)かつオフセットが92以上である場合、引数の第7ワードを検出したことを記憶する。

【0125】

その後、RWのTOPからBOTTOMまで順にたどる。そして、(a) アドレスが、RBに登録されているsp+64の場合、構造体ポインタの読み出しであるので、arg0.V=0であれば、arg0.V=2に変更する。(b) 上記の(a)の場合ではなく、アドレスがLIMIT以上sp+92未満であれば、登録不要領域であるので、何もしない。(c) 上記の(b)の場合でない場合、WRITEデータとして登録済であるかどうかを検査し、そうであれば、すでにアドレスは登録済であるので、内容を新しいWRITEデータに更新する。(d) 上記の(c)でない場合、WRITEデータとしての登録が必要であるので、RFに主記憶WRITEアドレスを確保し、WRITEデータとして登録する。RFに主記憶アドレスを確保できなかった場合には、登録不能であるため、そのRWエントリからBOTTOMまでに対応するRBエントリを全て無効化する。

20

30

【0126】

(並列事前実行)

以上に述べた、関数やループの多重再利用では、RBエントリの生存時間よりも同一パラメータが出現する間隔が長い場合や、パラメータが単調に変化し続ける場合には全く効果がないことになる。すなわち、RBエントリの生存時間よりも同一パラメータが出現する間隔が長い場合には、ある関数またはループがRBに登録されたとしても、その登録された関数またはループに関して同一パラメータが次に出現した際には、すでにその関数またはループがRBエントリから消えていることになり、再利用できないことになる。また、パラメータが単調に変化し続ける場合には、該当する関数やループがRBに登録されていても、パラメータが異なることによって再利用できないことになる。

40

【0127】

これに対して、多重再利用を行うプロセッサとしてのMSP1Aとは別に、命令区間の事前実行によってRBエントリを有効にするプロセッサとしてのSSP1Bを複数個設けることによって、さらなる高速化を図ることができる。

【0128】

並列事前実行機構を行うためのハードウェア構成は、前記した図5に示すような構成となる。同図に示すように、RW4A・4B、演算器5A・5B、レジスタ6A・6B、キャッシュ7A・7Bは、各プロセッサごとに独立して設けられている一方、RF/RB2、および主記憶3は全てのプロセッサが共有するようになっている。同図において、破線は、MSP1AおよびSSP1BがRF/RB2に対して入出力を登録するパスを示している

50

る。

【0129】

ここで、並列事前実行を実現する上での課題は、(1) どのように主記憶一貫性を保つか、(2) どのように入力を予測するか、(3) どのようにRBエントリを入れ替えるか、(4) どの命令区間を実行するか、の4点が挙げられる。以下に、これらの課題に対する解決手法について説明する。

【0130】

(主記憶一貫性に関する課題の解決方法)

まず、上記の課題(1) どのように主記憶一貫性を保つかについて説明する。特に予測した入力パラメータに基づいて命令区間を実行する場合、主記憶に書き込む値がMSP1AとSSP1Bとで異なることになる。これを解決するために、図5に示すように、SSP1Bは、RBへの登録対象となる主記憶参照にはRF/RB2、また、その他の局所的な参照にはSSP1Bごとに設けた局所メモリとしてのLocal7Bを使用することとし、Cache7Bおよび主記憶3への書き込みを不要としている。なお、MSP1Aが主記憶3に対して書き込みを行った場合には、対応するSSP1Bのキャッシュラインが無効化される。

10

【0131】

具体的には、RBへの登録対象のうち、読み出しが先行するアドレスについては主記憶3を参照し、MSP1Aと同様にアドレスおよび値をRBへ登録する。以後、主記憶3ではなくRBを参照することによって、他のプロセッサからの上書きによる矛盾の発生を避けることができる。局所的な参照については、読み出しが先行するということは、変数を初期化せずに使うことに相当し、値は不定でよいことになるので、主記憶3を参照する必要はない。

20

【0132】

なお、局所メモリとしてのLocal7Bの容量は有限であり、関数フレームの大きさがLocal7Bの容量を超えた場合など、実行を継続できない場合は、事前実行を打ち切るようにする。また、事前実行の結果は主記憶3に書き込まれないので、事前実行結果を使って、さらに次の事前実行を行うことはできない。

【0133】

(入力の予測方法)

次に、上記の課題(2) どのように入力を予測するかについて説明する。事前実行に際しては、RBの使用履歴に基づいて将来の入力を予測し、SSP1Bへ渡す必要がある。このために、RFの各エントリごとに小さなプロセッサを設け、MSP1AやSSP1Bとは独立して入力予測値を求めるようにする。

30

【0134】

具体的には、最後に出現した引数(B) および最近出現した2組の引数の差分(D)に基づいて、ストライド予測を行う。なお、B+Dに基づく命令区間の実行はMSP1Aがすでに開始していると考え、SSP1BがN台の場合には、用意する入力予測値は、B+D×2からB+D×(N+1)までの範囲とする。

【0135】

以上のように入力予測を行えば、上記した入力パラメータが単調に変化し続けるような場合に、事前に予測しておいた結果に基づいて効果的に再利用を行うことが可能となる。

40

【0136】

(RBエントリ入れ替え方法)

次に、上記の課題(3) どのようにRBエントリを入れ替えるかについて説明する。各RFエントリが1つの命令区間に対応し、入力と出力との対応関係がRBに登録される際に、MSP1AとSSP1BとがRBエントリをどのように使い分けるかが課題となる。命令区間は、大きく分けると、MSP1Aのみでも再利用の効果があるものと、配列を扱うループのようにMSP1Aのみでは効果がないものとに分けられると考えられる。前者であれば、LRU(least recently used)による入れ替え、後者であ

50

れば、F I F O (F i r s t I n F i r s t O u t) による入れ替えが有効である。

【0137】

しかしながら、ある命令区間の性質がいずれであるかを動的かつ直ちに判断することは難しいので、個々のR Fに属するR BエントリをM S P 1 A用とS S P 1 B用とに分割し、M S P 1 A用のR BエントリをL R Uによって、S S P 1 B用のR BエントリをF I F Oによって入れ替えるようにする。前述したように、入力予測値はN組であり、S S Pが登録後、M S Pが直ちに利用することを想定して、S S P用に割り当てるエントリ数は $N \times 2$ としておく。この様子を図6に示す。

【0138】

(命令区間の選択)

次に、上記の課題(4)どの命令区間を実行するかについて説明する。事前実行は、前記したように、同一パラメータが出現する間隔が長い命令区間や、パラメータが単調に変化し続ける命令区間に対して効果があることが予想される。しかしながら、それぞれの命令区間の性質や実際の効果の有無は、事前に認識することはできない。そこで、R Fに新規に登録された命令区間については、直ちにS S P 1 Bによる数回分の事前実行を試みるようにする。そして、数回の試行の結果、M S P 1 Aによる登録頻度が高く、かつ、S S P 1 Bが登録したエントリの再利用頻度も高いR Fを継続してS S P 1 Bの実行対象とする。

【0139】

具体的には、動的に変化する登録頻度や再利用頻度を把握するために、R Fにおける各エントリごとに、一定期間における登録および再利用の状況をシフトレジスタに記録する。R Fごとに付加した小さなプロセッサが、 $E = (\text{過去の削減ステップ数}) \times (\text{登録回数}) \times (\text{再利用回数})$ を計算し、各S S Pが、Eが最大となるR Fを選択する。この様子を図7に示す。以下に、このことについてより詳しく説明する。

【0140】

事前実行するかどうかを判断する際に利用できる統計情報としては、(a) 再利用によって削減可能なステップ数、(b) M S P 1 Aが、やむをえず実行し登録した頻度、(c) M S P 1 Aが、S S P 1 Bの登録結果を再利用できた頻度、(d) M S P 1 Aが、M S P 1 Aの登録結果を再利用できた頻度、が挙げられる。これらから、次にどの命令区間を事前実行すれば最大の効果が期待できるかを考える。

【0141】

まず、上記(d) M S P 1 Aが、M S P 1 Aの登録結果を再利用できた頻度に関しては、S S P 1 Bによる事前実行とは無関係であるので、考慮する必要はないことがわかる。

【0142】

上記(a)、(b)および(c)に関しては、自項目以外の項目が同じであるならば、それぞれ値が大きいほど効果を期待できることになる。また、それぞれの項目において、値が0であれば、効果を期待できないことになる。以上より、前記した式 $E = (\text{過去の削減ステップ数}) \times (\text{登録回数}) \times (\text{再利用回数})$ によって期待値を算出している。

【0143】

(本発明の適用例)

「L I M I T」などによって大域変数領域とスタック領域とを区別できるプログラム実行環境があるとした上で、本発明に係るデータ処理装置を他の命令セットアーキテクチャにも適用するためには、スタックフレーム上の変数が、上位/下位関数のどちらの局所変数であるかを区別する手段が必要である。特に、引数を格納するレジスタが不足し、引数をスタックに格納する場合、呼ばれた関数側ではこの区別をすることができないことになる。

【0144】

本実施の形態で取り上げたS P A R Cプロセッサでは、引数の先頭6ワードを汎用レジスタに格納しており、6ワード以上の引数を扱う関数は出現頻度が低いことと、引数が

10

20

30

40

50

スタックに溢れた時点で再利用ができなくなることの両方を利用することによって、関数／ループの再利用を実現している。SPARCプロセッサ同様に、32本以上の汎用レジスタを有する多くのRISCプロセッサでも、同様の判断をすることによって、本発明のような関数／ループの再利用を実現することが可能である。

【0145】

【発明の効果】

以上のように、本発明に係るデータ処理装置は、上記主記憶手段から読み出した命令列をデコードし、該命令列に基づく演算を行う第1の演算手段と、1つ以上の命令列に関する情報を記憶する命令列記憶手段とを備え、上記第1の演算手段が、命令列をデコードした結果、該命令列が後方分岐命令に挟まれたループであるか否かを判断する第1のステップと、上記第1のステップにおいて、ループであると判断された場合に、該ループに関する情報が上記命令列記憶手段に記憶されているか否かを判断する第2のステップと、上記第2のステップにおいて、記憶されていないと判断された場合に、該ループを実行する演算を行うとともに、上記命令列記憶手段に対して、演算結果に基づいて該ループに関する入力および出力を記録する第3のステップと、上記第2のステップにおいて、記憶されていると判断された場合に、処理対象としてのループに対する入力と、上記命令列記憶手段に記憶されているループに対する出力とが一致している場合に、上記命令列記憶手段に記憶されているループに対する出力の値を、処理対象としてのループに対する出力として出力する第4のステップとを行う構成である。

【0146】

これにより、あるループに関する命令列が複数回行われる場合には、2回目以降では、第1の演算手段による演算処理を行うことなく、命令列記憶手段に記憶されている出力を読み出すという処理のみで対応することが可能となる。よって、処理時間の大幅な短縮を実現することができるという効果を奏する。

【0147】

また、本発明に係るデータ処理装置は、上記第1のステップにおいて、さらに、命令列をデコードした結果、該命令列が関数であるか否かも判断し、上記第1のステップにおいて、関数であると判断された場合に、該関数に関する情報が上記命令列記憶手段に記憶されているか否かを判断する第5のステップと、上記第5のステップにおいて、記憶されていないと判断された場合に、該関数を実行する演算を行うとともに、上記命令列記憶手段に対して、演算結果に基づいて該関数に関する入力および出力を記録する第6のステップと、上記第5のステップにおいて、記憶されていると判断された場合に、処理対象としての関数に対する入力と、上記命令列記憶手段に記憶されている関数に対する入力とが一致している場合に、上記命令列記憶手段に記憶されている関数に対する出力の値を、処理対象としての関数に対する出力として出力する第7のステップとを行う構成としてもよい。

【0148】

これにより、上記の構成による効果に加えて、命令列が関数である場合にも、上記のループと同様に値再利用を実現することができるので、処理時間の大幅な短縮を実現することができるという効果を奏する。

【0149】

また、本発明に係るデータ処理装置は、上記第3のステップにおいて、上記命令列記憶手段に対して、主記憶手段における該ループの開始アドレスおよび終了アドレスと、全ての入出力に関する主記憶アドレスとを記録するとともに、上記第4のステップにおいて、処理対象としてのループの開始アドレス、ならびに、入力に関する主記憶アドレスが、命令列記憶手段に記憶されているものと完全に一致した場合に、上記命令列記憶手段に記憶されているループに対する出力の値を、処理対象としてのループに対する出力として出力する構成としてもよい。

【0150】

これにより、上記の構成による効果に加えて、的確にループを特定することが可能となり、ループに関する再利用を実現することが可能となるという効果を奏する。

【0151】

また、本発明に係るデータ処理装置は、上記第6のステップにおいて、上記命令列記憶手段に対して、主記憶手段における該関数の先頭アドレスと、入出力に関する主記憶アドレスとを記録するとともに、上記第7のステップにおいて、処理対象としての関数の先頭アドレス、ならびに、入力に関する主記憶アドレスが、命令列記憶手段に記憶されているものと完全に一致した場合に、上記命令列記憶手段に記憶されている関数に対する出力の値を、処理対象としての関数に対する出力として出力する構成としてもよい。

【0152】

これにより、上記の構成による効果に加えて、的確に関数を特定することが可能となり、関数に関する再利用を実現することが可能となるという効果を奏する。

10

【0153】

また、本発明に係るデータ処理装置は、上記主記憶手段から読み出された入力を一時的に格納し、上記第1の演算手段にわたすとともに、上記第1の演算手段からの出力を一時的に格納し、上記主記憶手段にわたすレジスタをさらに備えるとともに、上記命令列記憶手段に、ループの入力および出力が格納される全てのレジスタの情報が記録される構成としてもよい。

【0154】

これにより、上記の構成による効果に加えて、第1の演算手段は、レジスタの値に基づいて、再利用が可能か否かの判断を行うことができるので、この判断を的確かつ迅速に行うことが可能となるという効果を奏する。

20

【0155】

また、本発明に係るデータ処理装置は、特定の上位の命令区間が、その内部に1つ以上の下位の命令区間を含んでいる場合に、下位の命令区間の実行中に、該下位の命令区間が行った入出力が上位の命令区間の局所変数である場合には、該下位の命令区間の入出力として上記命令列記憶手段に登録し、該下位の命令区間が行った入出力が上位の命令区間の局所変数でない場合には、該下位の命令区間および上位の命令区間の入出力として上記命令列記憶手段に登録する構成としてもよい。

【0156】

これにより、上記の構成による効果に加えて、例えば命令列記憶手段に上位の命令区間に関する情報が登録されている場合に、該命令区間が出現した場合に、その入力を参照することによって、下位の命令区間を考慮することなく、的確に該命令区間の再利用を実現することが可能となるという効果を奏する。

30

【0157】

また、本発明に係るデータ処理装置は、上記命令列記憶手段に対して、上位の命令区間および1つ以上の下位の命令区間が登録されている場合に、登録されている命令区間の包含関係の順番をスタック構造で記憶する再利用記憶手段をさらに備えている構成としてもよい。

【0158】

これにより、上記の構成による効果に加えて、命令列記憶手段に多重入出力構造の命令区間が複数登録されていても、再利用記憶手段を参照することによって、各命令区間の包含関係を認識することが可能となるという効果を奏する。

40

【0159】

また、本発明に係るデータ処理装置は、上記再利用記憶手段に登録可能な命令区間の多重度を超えて、命令区間を検出した場合、その時点で最も上位の命令区間の登録を中止する構成としてもよい。

【0160】

これにより、上記の構成による効果に加えて、再利用可能な多重入出力構造の登録を最低限確保することが可能となるという効果を奏する。

【0161】

また、本発明に係るデータ処理装置は、少なくとも1つの第2の演算手段をさらに備え、

50

上記第2の演算手段が、上記第1の演算手段によって処理が行われている命令列に関して、今後入力が予想される予測入力値に基づいて該命令列の演算を行い、その結果を上記命令列記憶手段に対して登録する構成としてもよい。

【0162】

これにより、上記の構成による効果に加えて、次に、同じ命令列が出現し、予測入力値と同じ入力が行われた場合には、命令列記憶手段に記憶されている値を再利用することが可能となるという効果を奏する。

【0163】

また、本発明に係るデータ処理装置は、上記第1の演算手段に対して、主記憶手段に対して読み出しおよび書き込みが可能なキャッシュメモリが設けられているとともに、上記第2の演算手段に対して、主記憶手段に対して読み出しのみ可能なキャッシュメモリが設けられており、上記第2の演算手段が、命令列記憶手段に対する入出力記録対象となる主記憶参照には命令列記憶手段に記録されている入出力記録そのものを用い、その他の局所的な参照には、上記第2の演算手段に設けられたローカルメモリを用いるとともに、上記第1の演算手段が主記憶手段へ書き込みを行う場合には、対応する第2の演算手段のキャッシュラインが無効化される構成としてもよい。

【0164】

これにより、上記の構成による効果に加えて、第1の演算手段と、第2の演算手段とで、主記憶手段に対して異なる値を書き込むことによる不具合を防止することができるという効果を奏する。

【0165】

また、第1の演算手段によって主記憶手段が書き換えられた場合にも、第2の演算手段が、新たに主記憶手段から値を読み出すことによって、主記憶一貫性を保つことができるという効果を奏する。

【0166】

また、本発明に係るデータ処理装置は、上記命令列記憶手段において、上記第1の演算手段による書き込み範囲と、上記第2の演算手段による書き込み範囲とを分割するとともに、上記第1の演算手段による書き込み範囲では、LRU (least recently used) によってエントリの入れ替えが行われるようにし、上記第2の演算手段による書き込み範囲では、FIFO (First In First Out) によってエントリの入れ替えが行われるようにする構成としてもよい。

【0167】

これにより、上記の構成による効果に加えて、エントリの入れ替えは、第1の演算手段による書き込み範囲ではLRU、第2の演算手段による書き込み範囲ではFIFOとしているので、より効率的に再利用を行うことが可能となるという効果を奏する。

【0168】

また、本発明に係るデータ処理装置は、命令列記憶手段における各エントリごとに、一定期間における登録および再利用の状況が記録され、 $E = (\text{過去の削減ステップ数}) \times (\text{登録回数}) \times (\text{再利用回数})$ が算出されるとともに、上記第2の演算手段が、上記Eが最大となるエントリに関して事前実行を行う構成としてもよい。

【0169】

これにより、上記の構成による効果に加えて、第1の演算手段による登録頻度が高く、かつ、第2の演算手段が登録したエントリの再利用頻度も高い命令列を継続して第2の演算手段の実行対象とすることができる。よって、第2の演算手段による事前実行の効果を最大限高めることが可能となるという効果を奏する。

【図面の簡単な説明】

【図1】本発明の一実施形態に係るデータ処理装置が備えるRF/RBによって実現される再利用表を示す図である。

【図2】同図(a)は、関数が呼び出されて実行される様子を概念的に示す図であり、同図(b)は、ループ処理が実行される様子を概念的に示す図である。

10

20

30

40

50

【図 3】関数の入れ子構造において、内側の構造のレジスタ入出力が、外側の構造のレジスタ入出力となる影響範囲を示す図である。

【図 4】RW と、RF・RB との関係を示す図である。

【図 5】上記データ処理装置の概略構成を示すブロック図である。

【図 6】RB エントリを分割する状態を示す図である。

【図 7】RF における各エントリごとに、一定期間における登録および再利用の状況をシフトレジスタに記録する様子を示す図である。

【図 8】関数およびループが入れ子構造となっている状態の一例を示す図である。

【図 9】命令がデコードされた結果、関数呼び出し命令である場合の処理の流れを示すフローチャートである。

10

【図 10】命令がデコードされた結果、関数復帰命令である場合の処理の流れを示すフローチャートである。

【図 11】命令がデコードされた結果、後方分岐成立である場合の処理の流れを示すフローチャートである。

【図 12】命令がデコードされた結果、後方分岐不成立である場合の処理の流れを示すフローチャートである。

【図 13】同図 (a) は、関数 A が関数 B を呼び出す構造を概念的に示す概念図であり、同図 (b) は、同図 (a) に示すプログラム構造を実行する際の主記憶におけるメモリマップを示す図である。

【図 14】関数 A が関数 B を呼び出す場合の、メモリマップにおける引数およびフレームの概要を示す図である。

20

【図 15】1つの関数を再利用するための従来の再利用表を示す図である。

【符号の説明】

1 A MSP

1 B SSP

2 RF / RB (命令列記憶手段)

3 主記憶 (主記憶手段)

4 A・4 B RW (再利用記憶手段)

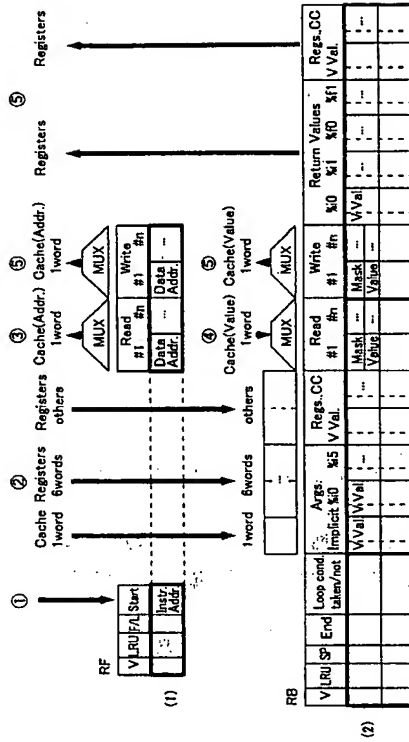
5 A・5 B 演算器 (第 1・第 2 の演算手段)

6 A・6 B レジスタ

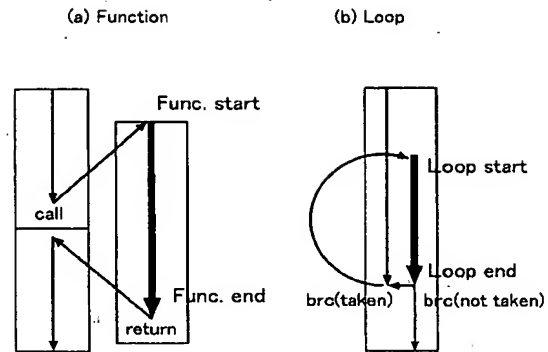
7 A・7 B Cache

30

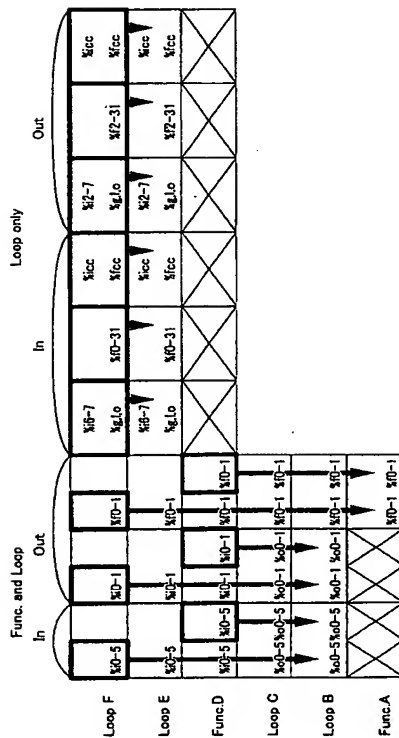
【図 1】



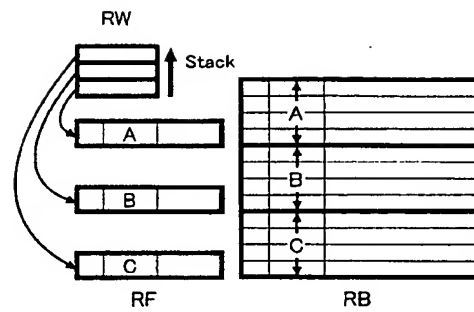
【図 2】



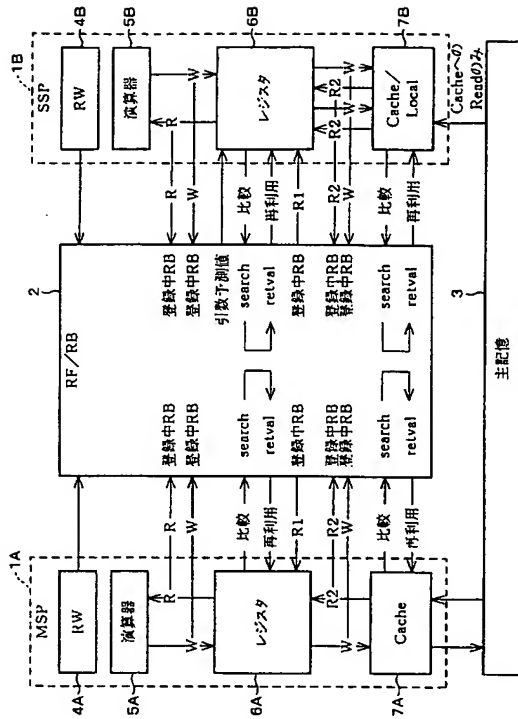
【図 3】



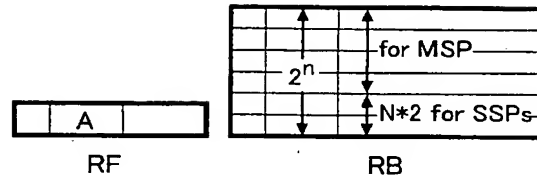
【図 4】



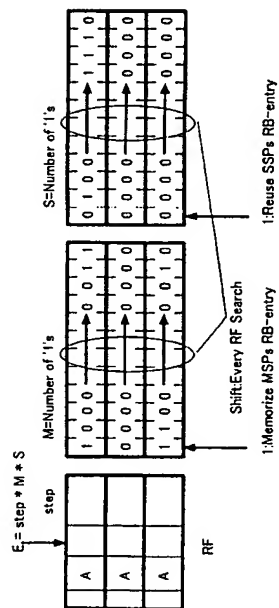
【図 5】



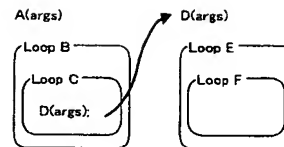
【図 6】



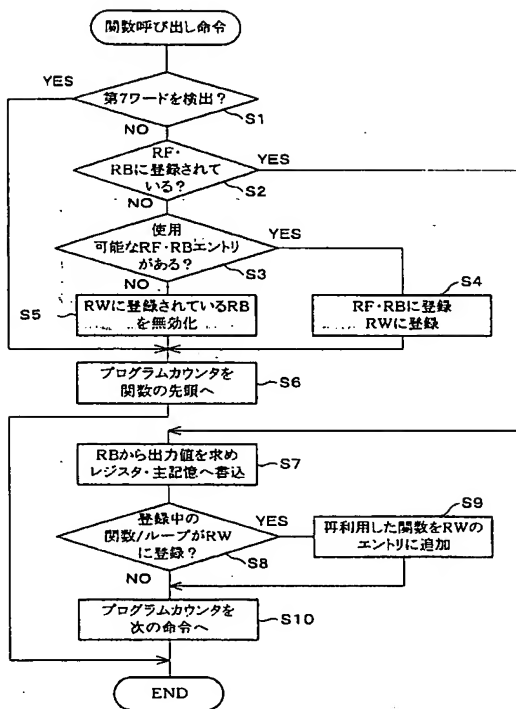
【 図 7 】



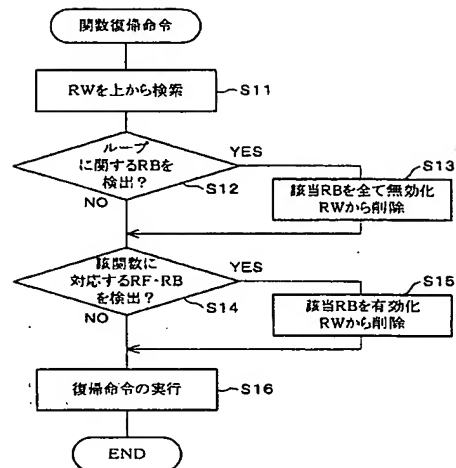
【图 8】



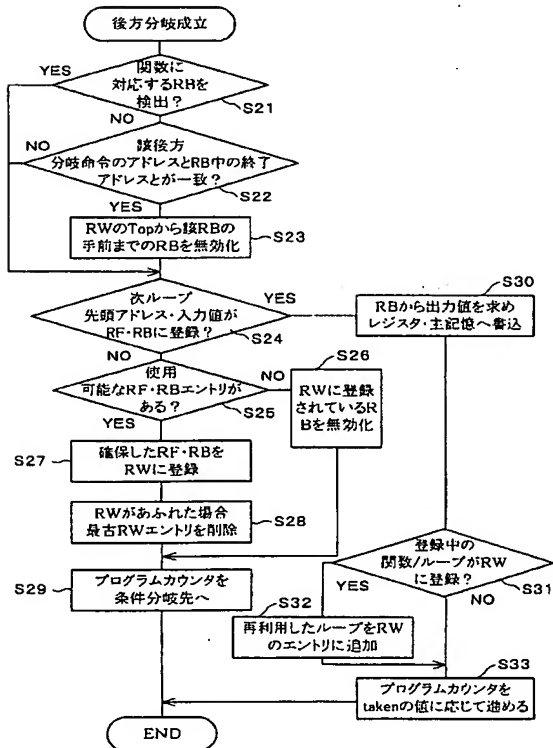
【図 9】



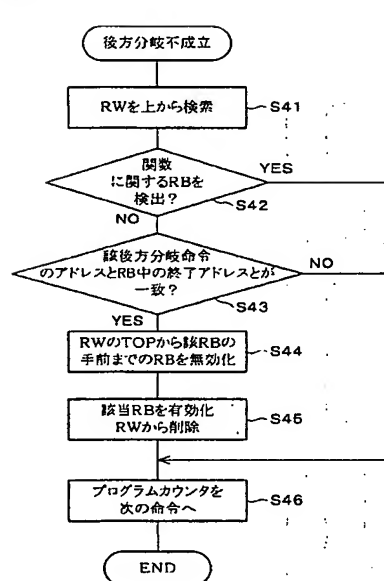
【図 10】



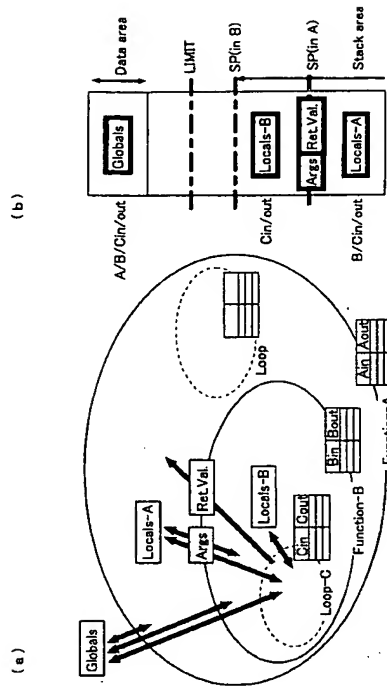
【図 11】



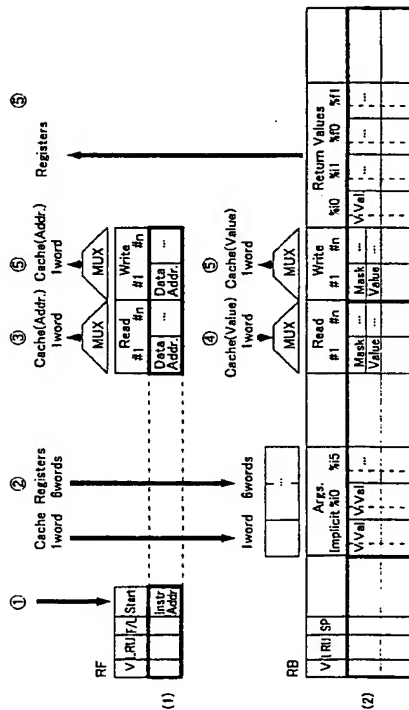
【図 12】



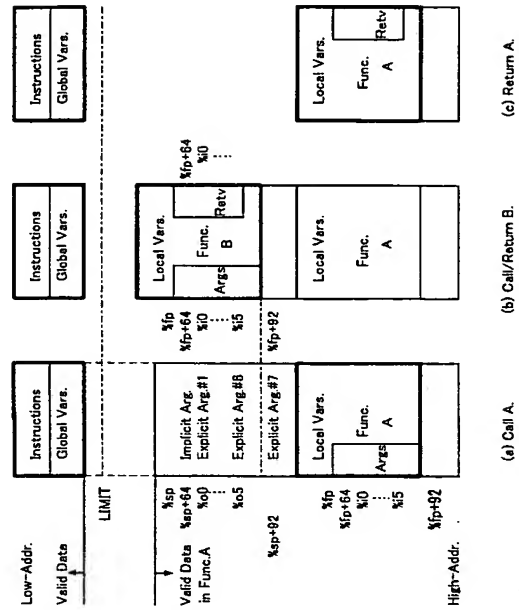
【 図 1 3 】



【 図 1 5 】



【 図 1 4 】



**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☒ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.